PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/50, 159/00, G06G 7/58, G01N 24/12, 33/53	A1	(11) International Publication Number: WO 96/30849 (43) International Publication Date: 3 October 1996 (03.10.96)

(21) International Application Number: PCT/US96/04229

(22) International Filing Date: 27 March 1996 (27.03.96)

(30) Priority Data:
418,992 31 March 1995 (31.03.95) US

(71) Applicant: CURAGEN CORPORATION [US/US]; 322 East Main Street, Branford, CT 06405 (US).

(72) Inventors: DEEM, Michael, W.; 24 Highland Avenue #14, Cambridge, MA 02139 (US). ROTHBERG, Jonathan, M.; 49 Old Quarry Road, Guilford, CT 06437 (US). WENT, Gregory, T.; 34 Scotland Avenue, Madison, CT 06443 (US).

(74) Agents: MORRIS, Francis, E. et al.; Pennie & Edmonds, 1155 Avenue of the Americas, New York, NY 10036 (US).

(81) Designated States: AL, AM, AU, AZ, BB, BG, BR, BY, CA, CN, CZ, EE, FI, GE, HU, IS, JP, KG, KP, KR, KZ, LK, LR, LS, LT, LV, MD, MG, MK, MN, MX, NO, NZ, PL, RO, RU, SG, SI, SK, TJ, TM, TR, TT, UA, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: CONSENSUS CONFIGURATIONAL BIAS MONTE CARLO METHOD AND SYSTEM FOR PHARMACOPHORE STRUCTURE DETERMINATION

(57) Abstract

In a specific embodiment, this invention comprises a method for selecting highly targeted lead compounds for design of a drug that binds to a target molecule. The method comprises screening a diversity library against the target molecule of interest to pick the selectively binding members. Next the structure of the selected members is examined and a candidate pharmacophore responsible for the binding to the target molecule is determined. Next, preferably by REDOR nuclear magnetic resonance, several highly accurate interatomic distances are determined in certain of the selected members which are related to the candidate pharmacophore. A highly accurate consensus, configurational bias, Monte Carlo method determination of the structure of the candidate pharmacophore is made using the structure of the selected members and incorporating as constraints the shared selected members and incorporating as constraints the shared candidate phamacophore and the several measured distances. This determination is adapted to efficiently examine only relatively low energy configurations while respecting any structural constraints present in the organic diversity library. If the diversity library contains short peptides, the determination respects the known degrees of freedom of peptides as well as any internal constraints, such as those imposed by disulfide bridges. Finally, the highly accurate pharmacophore so determined is used to select lead organics for drug development targeted at the initial target molecule.

CONSENSUS CONFIGURATIONAL BIAS MONTE CARLO METHOD AND SYSTEM FOR PHARMACOPHORE STRUCTURE DETERMINATION

This specification includes in Sec. 8 computer program listings that are exemplary embodiments of the computer programs of this invention.

This invention was made with Government support under Grant number 1R43CA62752-01 awarded by the National Institutes of Health. The Government has certain rights in the invention.

1. FIELD OF THE INVENTION

The field of this invention is computer assisted methods of drug design. More particularly the field of this invention is computer implemented smart Monte Carlo methods which utilize NMR and binders to a target of interest as inputs to determine highly accurate molecular structures that must be possessed by a drug in order to achieve an effect of interest. Illustrative U.S. Patents are 5,331,573 to Balaji et al., 5,307,287 to Cramer, III et al., 5,241,470 to Lee at al., and 5,265,030 to Skolnick et al.

2. BACKGROUND

Protein interactions have recently emerged as a fundamental target for pharmacological intervention. For example, the top two major uncured diseases in the United States are atherosclerosis (the principal cause of heart attack and stroke) and cancer. These diseases are responsible for greater than 50% of all U.S. mortality and cost the U.S. economy over $200 billion per year. A consistent picture of these dis ases, which has gradually emerged during the past ten y ars of molecular biological and medical research, views both as triggered by disordering of specific molecular recognition events that take place among sets of proteins present in both the normal and disease states.

Hierarchical, organized patterns of protein-protein interactions are often referred to as "pathways" or "cascades." At the molecular level, cancers have been determined to be the deregulation of pathways of interacting proteins responsible for guiding cellular growth and differentiation. During the past year, individual cellular events have been organized into nearly complete mechanistic explanations of how a cell's behavior is controlled by its environment and how communication pathway errors lead to uncontrolled proliferation and cancer. Disruption in similar pathways are responsible for the proliferation of blood vessel walls marking the atherosclerotic disease state (Cook et al., 1994, Nature 369:361-362; Hall, 1994, Science 264:1413-1414; Ross, 1993, Nature 362:801-809; Zhang et al., 1993, Nature 364:308-313).

Inhibition or stimulation of particular protein-substrate interactions have long been known drug targets. Many important anti-hypertensives, neurotransmitter analogues, antibiotics, and chemotherapeutic agents act in this fashion. Captopril, an antihypertensive drug, was designed based on its ability to antagonize a focal blood-pressure-regulating enzyme.

Proteins involved in biological processes, either as part of protein-protein pathways or as enzymes, are composed of domains (Campbell et al., 1994, Trend. BioTech. 12:168-172; Rothberg et al., 1992, J. Mol. Biol. 227:367-370). Domains, or regions of the protein of stable three dimensional (secondary and t rtiary) structures, play several major roles, including providing on their surface small regions ("examples of targ ts"), wher proteins and substrat s are abl to bind and interact, and functioning as structural units holding other domains together as part of a large protein (tertiary and quaternary structure). The interaction surface of a domain or target is fundamental to determining binding specificity. Targets are often small enough that the principal contribution to the binding energy is short range, highly localized to several amino acids (Wells, 1994, Curr. Op. Cell Biol. 6:163-174). The functional specificity of targets and domains, responsible for the incredible diversity of cellular function, ultimately rests with the arrangement of amino acid side chains forming their interaction surfaces, or targets (Marengere et al., 1994, Nature 369:502-505).

It can be appreciated, therefore, that pharmacological intervention affecting the specific protein-protein and protein-substrate recognition events occurring at protein targets is of fundamental importance, particularly for effective drug design.

However, achieving desired pharmacological interventions in a predictable manner remains as elusive as ever. Early approaches to drug design depended on the chance observation of biological effects of a known compound or the screening of large numbers of exotic compounds, usually derived from natural sources, for any biological effects. The nature of the actual protein target was usually unknown.

2.1. TARGET STRUCTURE-BASED APPROACHES TO DRUG DESIGN

Rational approaches to drug design have met with only limited success. Current rational approaches are based on first determining the entire structure of the proteins involved in particular interactions, examining this structure for the possible targets, and then predicting possible drug molecules likely to bind to the possible target. Thus the location of each of the thousands of atoms in a protein must be accurately determined before drug d sign can begin.

Direct experimental and indirect computational methods for protein structure determination are in current use. However, none of these methods appears to be sufficiently accurate for drug design purposes according to current rational approaches.

The primary direct experimental methods for determining the structure of proteins involved in particular interactions are X-ray crystallography, relying on the interaction of electron clouds with X-rays, and liquid nuclear magnetic resonance (NMR), relying on correlations between polarized nuclear spins interacting via indirect dipole-dipole interactions. X-ray methods provide information on the location of every heavy atom in a crystal of interest accurate to 0.5-2.0 Å (1 Å = 10^{-8} cm). Drawbacks of x-ray methods include difficulties in obtaining high-quality crystals, expense and time associated with the crystallization process, and difficulties in resolving whether or not the structure of the crystalline forms is representative of the *in vivo* conformation (Clore et al., 1991, J. Mol. Biol. 221:47; Shaanan et al., 1992, Science 227:961-964). High resolution, multidimensional, liquid phase NMR techniques represent an attractive alternative, to the extent that they can be applied *in situ* (*i.e.*, in aqueous environment) to the study of small protein domains (Yu et al., 1994, Cell 76:933-945). However, the complexity of the analysis of the various mutual correlations is time consuming, and the correlations (primarily from the nuclear Overhausser effect) provide no better accuracy than X-ray methods. Isotopic enrichment of proteins with ^{13}C and ^{15}N reduces the time associated with analysis, but at a great expense (Anglister et al., 1993, Frontiers of NMR in Biology III LZ011).

Protein structures determined by any of these current methods do not predict success in subsequent drug design. Resolution obtainable either by measurement or computation, g nerally 0.5-2 Å, has oft n been found to be inadequate for effectiv direct drug design, or for selection of a lead compound from organic compound libraries. The resolution required to understand both drug affinity and drug specificity, although not precisely known, is probably measured in fractions of an Å, down to 0.1 Å (MacArthur et al., 1994, Trend. BioTech. 12:149-153). This accuracy appears to be beyond the capabilities of many current methodologies.

Prior research has identified tools which, although promising, cannot be used in a coordinated manner for drug design. One promising measurement approach with speed, simplicity, accuracy, and the ability to carefully control the measurement environment is rotational echo double resonance (REDOR) NMR, a type of solid state NMR (Guillion and Schaefer, 1989, J. Magnetic Resonance 81:196; Holl et al., 1990, J. Magnetic Resonance 81:620-626 and McWherter, 1993, J. Am. Chem. Soc. 115:238-244). REDOR accuracy can be below the 0.1 Å believed to be sufficient for direct drug design. However, since REDOR measures only a few selected distances, it is not usable in drug design methods which depend on the initial determination of the complete structure of the protein containing the target of interest.

Once a target's structure is determined by the above methods, most rational drug design paradigms call for the prediction of small drug structures that will bind (or dock) to the target. This prediction is generally done by computational methods, of which several are in current use. Most seek to predict the position of all the thousands of atoms in a drug structure. Purely *ab initio* computational approaches to high resolution structure analysis, such as quantum statistical mechanics and molecular dynamics, require prohibitive computing resources. To apply either approach, the potential energy, or Hamiltonian, of the entire system must be known. Statistical mechanics provides an expression for the probability of any given protein configuration as a ratio of partition functions. Proper quantum statistical mechanics required for an exact evaluation of full protein partition functions is not curr ntly computationally feasibl , as it would involve many thousands of atoms including the target, the protein, and the aqueous environment. The application of even simple, approximate quantum statistical mechanics to simple systems in aqueous environments is currently a non-trivial task (Chandler, 1991, in Liquids, Freezing, and Glass Transitions, Elsevier, NY, p. 195). Molecular dynamics computes the dynamics of a molecule's motion in time. Computing the atomic dynamics of all the perhaps thousands atoms of a protein is an extreme computational burden. Only picoseconds, or at most a few nanoseconds, of molecular time can be simulated, which is insufficient to determine a high resolution, equilibrium, structure (Smit et al., 1994, J. Phys. Chem. 98:8442-8452). In any case, most of the information determined is wasted, since only the structure of the protein binding target are of interest in drug design.

Further, current approximate computational techniques for protein structure determination are in need of greater accuracy or efficiency. The most common techniques depend on Molecular Dynamics or Monte Carlo methods (Nikiforovich, 1994, Int. J. Peptide Protein Res. 44:513-531; Brünger and Karplus, 1991, Acc. Chem. Res. 24:54-61). These methods randomly alter initial molecular structures by generating simulated thermal perturbations, and then average the ensemble of results to determine a final structure. The generated perturbation must preserve all structural constraints and be energetically favorable. If both conditions are not met, the perturbation will be discarded. Current Monte Carlo methods applied to constrained protein structure determinations productively use only approximately 1 out of 10^5 perturbed structures generated (Siepmann et al., 1993, Nature 365:330-332). This extreme waste of computer rescurces results in time consuming, low resolution structure determinations.

To summarize, existing rational drug design methods bas d on identification of target structure fail to r liably yield drug molecules du to experimental structure determination difficulties and computational difficulties associated with predicting drug structures with ill-defined Hamiltonians.

2.2. DIVERSITY-BASED APPROACHES TO DRUG DESIGN

Another method for exploring protein target interactions utilizes "recognition systems" which comprise huge libraries of related molecules (Clarkson et al., 1994, Trend. BioTech. 12:173-184). From such a library only those members binding to the target of interest are selected. Such recognition systems must encompass the structural diversity of protein targets while being amenable to serve for the selection of lead compounds for drug design. Antibodies are one classic example of such a system that certainly meets the recognition requirement. Unfortunately, there is a need to determine the antibody structures needed for lead compound selection more rapidly and accurately. While about 2000 recognition regions have been sequenced, only about 23 in the Brookhaven Protein Structural Database have structures determined to even within 2 Å (Rees et al., 1994, Trends in Biotech. 12:199-206).

Promising recognition systems at the opposite extreme comprise huge libraries of small peptides. The small peptides must be sufficiently diverse so that they attain a level of affinity and specificity similar to that obtained by protein domains. Given the role peptides play in nature, this condition can be met by surprisingly small structures, with 6 to 12 amino acids. However, linear peptides are either unstructured or weakly structured at room temperature in aqueous solutions (Alberg et al., 1993, Science 262:248; Skalicky et al., 1993, Protein Science 10:1591-1603). From a practical viewpoint, linear peptides must be constrained to reduce their degrees of freedom (reduced conformational entropy) and to increase their chances for strongly binding. These constraints, or scaffolds, limit the range of stable conformations and make more straightforward determining bound structure (Olivera et al., 1990, Science 249:259; Tidor et al., 1993, Prot ins: Structure Function and Genetics 15:71).

Methods are now available to create such libraries and to sel ct library members that recognize a specific protein target. The production of constrained peptide diversity libraries requires synthesizing oligonucleotides with the desired degeneracy to code for the peptides and ligating them into selection vectors (Goldman et al., 1994, Bio/Tech. 10:1557-1561). Once a constrained structured diversity library is created, it is a source from which to select specific members that bind to a target of interest. Beginning with a known pathway involving specific domain-domain or protein-substrate interactions at a target, molecular biological methods can be used to identify in a matter of days small ensembles of highly constrained peptides from these huge libraries that bind to these domains with high affinity and specificity.

While this field has been exploding in the last few years and showing great potential, it is severely limited by its use in isolation without the benefit of integrated structural analysis needed both to derive the high resolution structures of binding peptides and also to direct the construction of additional structured libraries. Drug design is not aided by having library members recognizing the protein target of interest but without any understanding of why the recognition occurs. This is entirely similar to the random screening methods of early fortuitous drug design efforts.

Unfortunately, rational drug design according to current approaches (target structure-based) remains an inefficient, laborious process with a disproportionately high lead-compound failure rate. Presently, about 90% of lead compounds fail to emerge successfully from clinical trials (Trends in U.S. Pharmaceutical Sales and Research and Development, Pharmaceutical Manufacturing Association, Washington, D.C., 1993).

It is becoming clear that low-resolution structures of an entire protein or target (at 0.5-2 Å), or an uncharacterized lead, such as produced by chemical diversity methods, leave much to be desired for use in drug design.

If the limitations of prior art methods were overcome and a sufficiently accurate structure needed by a molecule to bind to a target of interest could be determined, existing chemical libraries could be searched for highly targeted lead compounds with similar structure (Martin, 1992, J. Medicinal Chem. 35:2145-2154). This database search can be based not only on chemical and electronic properties, but also on geometric information. Such searches that have high resolution (better than 0.25 Å), would provide a vast improvement over the prior art, as lower resolutions lead to an exponentially increasing number of potential leads.

Computational methods to determine high resolution drug structures from recognition system binding information or NMR partial distance measurements are not currently available. No current structure determination methods uses such additional information to make more efficient or more accurate determination of high resolution structures (Holzman, 1994, Amer. Sci. 872:267).

Citation of a reference or discussion hereinabove shall not be construed as an admission that such is prior art to the present invention.

3. SUMMARY OF THE INVENTION

It is a broad object of this invention to address the prior art problems of drug design by providing a method of rational design of drugs that achieve their effect by binding to a target molecule or molecular complex of interest. Importantly, this object is achieved without requiring determination of the structure of the molecule or molecular complex ("target molecule") bearing the target or even of the target itself. The method is target structure independent. The method of the invention uses an interdisciplinary combination of computational modeling and simulation, experimental distance constraints, and molecular biology.

In an important aspect, the invention provides a computer implemented modeling and simulation method to determine a highly accurate consensus structure for the pharmacophore and a structure for the remainder of the molecule from diversity library members that bind to the protein target of interest. Where prior structure determination methods focused on the structure of the target molecule or of the target, the method of this invention is uniquely adapted to focus instead on the structures of molecules that bind to the target. Such structural information is directly applicable to drug design since it defines the structure a drug must possess to bind to the target of interest. Also, this structural information is much easier to determine by use of the present invention, since it concerns molecules with many fewer atoms than the target molecule. The method of the invention achieves accuracy by improving upon the accuracy and utility of the input structural information. In a further embodiment of the invention, the method employed for structural determination is a smart Monte Carlo technique adapted to small constrained molecules.

The structure determination method of the invention allows one to take maximum advantage of the information obtained from the molecular biological selection of the diversity library members that tightly and specifically bind to the target molecule of interest. The selected library members must share some common structure to bind to the same target molecule. The smart Monte Carlo computer method of this invention specifically seeks and provides this common structure.

The invention also provides a method of performing REDOR NMR measurements of molecules on a solid phase substrate. In a preferred embodiment, the substrate is a solid phase on which the molecule (*e.g.*, peptide) has been synthesized, with a high degree of purity. In another pr ferred mbodiment, performing REDOR measurements of such a molecule on a substrate can be done in a dry nitrogen atmospher , under hydrated conditions, and when the molecule is either free or bound to a target. In a specific embodim nt, the REDOR measurements are accurate to better than 0.05 Å from 0 to 4 Å, and to better than 0.1 Å from 4 to 8 Å. In an advantageous aspect of the invention, the structure determination method makes maximum use of these highly accurate internuclear distance measurements to constrain the determined common structure for the binding library members.

The invention also provides methods of identifying a compound that specifically binds to a target molecule, by first screening a diversity library, and then using a genetic selection method for screening the compounds identified from the diversity library.

In broad aspects, the invention provides a method and apparatus for rational and predictable design of new and/or improved drugs that achieve their effect by binding to a specified target molecule. More particularly, the invention is directed to a method for the rational selection of highly specific lead compounds for such drug design, including the computer implemented step of highly accurate determination of the structure responsible for this target binding by the highly accurate, consensus, configurational bias Monte Carlo method.

A lead compound serves as a starting point for drug development both because it specifically binds to the protein target of interest, achieving the biological effect of interest, and because it has or can be modified to have good pharmacokinetics and medicinal applicability. A final drug may be the lead compound or may be derived therefrom by modifying the lead to maximize beneficial effects and minimize harmful side-effects. Although any lead compound is useful, a lead that tightly and specifically binds to the target molecule of interest in a known manner, such as can be provided by the invention, is of great use. Knowledge of the high resolution structures in a lead compound responsible for its binding and activity provides a more focused and fficient drug dev lopment process.

The methods of the invention improve lead compound determination, by determining the "pharmacophore", the precise structural characteristics needed for a lead compound to specifically bind to a target of interest. The most fundamental specification of a pharmacophore is in terms of the electronic properties necessary for a molecule to specifically bind to the surface of a target molecule. These properties may be fundamentally represented by requirements on the ground and low lying excited state wave functions of a pharmacophore, such as, for example, by specifying requirements on the well known multiple expansion of these wave functions.

The preferred pharmacophore specification according to the invention is in terms of both the chemical groups making up the pharmacophore and determining its electronic properties and also the geometric relationships of these groups. This chemical representation is not the only possible representation of the pharmacophore. Several chemical arrangements may have similar electronic properties. For example, if a pharmacophore specification included an -OH group at a particular position, a substantially equivalent specification might include an -SH group at the same position. Equivalent chemical groups that may be substituted in a pharmacophore specification without substantially changing its nature are called "homologous".

In particular embodiments, therefore, this invention provides a method and apparatus for the highly accurate determination of the pharmacophore needed to specifically bind to the target molecule of interest, by a specification of the geometric relationships of the important chemical groups. The pharmacophore is preferably determined by a smart Monte Carlo method from molecular biological input specifying molecules (preferably selected from among diversity libraries) that specifically bind to the target molecule and also preferably from REDOR NMR data specifying a few highly accurate distances in these select d molecules.

An important advantage provided by the invention is the ability to make a pharmacophore structure determination without relying on any knowledge of the structure of the target molecule or target. Where the target molecule is a protein, conventional prior art methods have sought to sequence and determine the structure of the protein containing the target, hoping thereby to determine active sites by examination of the structure. A further important advantage of the invention is that this structure determination can be made by use of a relatively small number of actual physical position measurements. In contrast, conventional methods using X-ray crystallography and liquid NMR require determination of positions of all atoms in the molecule ("binder") that specifically binds to the target, and the target. An additional advantage provided by the invention is that, in a preferred embodiment wherein REDOR structural measurements provide input information, the accuracy of the pharmacophore structure determination can be at least approximately 0.25-0.50 Å or better. This accuracy is provided by the combination of an efficient, Monte Carlo technique for structure determination with a few highly accurate distance determinations.

4. BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become better understood by reference to the accompanying drawings, following description, and appended claims, where:

Fig. 1 is the overall method of this invention in its broadest aspect;

Fig. 2A and 2B are more detail for the step of Fig. 1 for selecting candidate pharmacophore structures;

Fig. 3 is more detail for the step of Fig. 1 for preforming distance measurements;

Fig. 4 is more detail for the step of Fig. 3 for performing NMR measur ments;

Fig. 5 is REDOR NMR signal response details for step of Fig. 3 of data analysis;

Fig. 6 is sample REDOR NMR spectra according to the method of Fig. 3;

Fig. 7 is sample data analysis according to the method of Fig. 3;

Fig. 8 is more detail for the Step of Fig. 1 for configurational bias Monte Carlo structure determination;

Fig. 9 is a sample of simulation completion data;

Fig. 10 is further detail of peptide memory representation used in the method of Fig. 8;

Fig. 11 is additional detail of peptide memory representation used in the method of Fig. 8;

Fig. 12 is more detail for the step of Fig. 8 of processor generation of proposed modified structures by Type I moves;

Fig. 13 is more detail for the step of Fig. 8 of processor generation of proposed modified structures by Type II moves;

Fig. 14 is additional detail for the step of Fig. 8 of processor generation of proposed modified structures by Type II moves;

Fig. 15 is a structure for implementing the method of Fig. 8;

Fig. 16 is the main program structure of Fig. 15;

Fig. 17 is the structure modification program structure of Fig. 15;

Fig. 18A and 18B are the Type I move generator program structure of Fig. 17;

Fig. 19A and 19B are the Type II move generator program structure of Fig. 17.

5. DETAILED DESCRIPTION

For clarity of disclosure, and not by way of limitation, the detailed description of the invention is described as a series of steps. A broad view of the ex mplary steps of which the invention is comprised is presented in Fig. 1, a brief overview of which is presented in the text that follows.

The invention method preferably begins with a target molecule (or molecular complex) 1 having a binding target of biological or pharmacological interest. Specific binding of a molecule to the target is predicted to affect its biological activity and may provide biological effects of interest. For example, these effects might include amelioration of a disease process or alteration of a physiological response. Lead compounds 8 output from the invention are able to specifically bind to target molecule 1 and can serve as starting points for the design of a drug able to specifically bind to the target.

Diversity library screening, step 2, allows the selection from among library members of a plurality of molecules [hereinafter called "binders"] that specifically bind to target molecule (or molecular complex) 1; the chemical building block structure (*e.g.*, sequence, structural formula) is then determined. If predetermined binders and their structure are already available, the invention can use this information directly without the need for library screening. If library screening is done, one or more libraries may be screened. The selected binders all share a common pharmacophore structure, allowing their specific binding to the target in a chemically and physically similar manner. This common structure is preferably iteratively determined by a select and test method. Candidate pharmacophore selection, step 3, is based upon chemical structure homologies. Geometric and conformational information is not needed to be used at this step and is preferably not considered. A candidate pharmacophore shared by all the N binders is selected, step 3, for structure determination by subsequent steps. The binders will typically present several candidate chemical pharmacophores, ignoring conformation considerations. These candidates are small groups of library building blocks, often contiguous, ach candidate group in on binder being homologous to the candidate groups in all the other binders. Building block homologies are determined by applying rules appropriat to the diversity library. In the preferred embodiment, homologous building blocks have similar surface chemical groups, since pharmacophores are defined by a similar geometric arrangement of chemical structures. In the case of the preferred library, CX_6C, candidate pharmacophores are amino acid sequences whose side chain surface groups have similar chemical properties. Amino acid homologies are determined by mechanical rules described below. These candidate sequences are typically 3 amino acids long, but may range from 2 all the way to 6. Where pharmacophores are defined by their charge distributions, homologous library building blocks must have similar charge distributions.

Having selected N binders by screening one or more libraries and determined a candidate pharmacophore in each binder, the subsequent steps of distance measurement, step 4, and Monte Carlo structure determination, step 5, determine a highly accurate structure for the candidate pharmacophore, if possible. This determination will be possible if the candidate is the actual pharmacophore. A subsequent test, step 6, checks for success of this structure determination. In particular cases, distance measurements may not be necessary in order to determine an adequately precise pharmacophore structure.

Measurements are made, step 4, of a few strategic distances in the binders, that will be most useful for the subsequent structure determination step. A minimum number of strategic interatomic distances in the binders are measured in step 4. These few distances constrain possible binder structures and make the subsequent complete structure determination more efficient and more accurate. In preferred but not limiting embodiments, measurement methods yielding distances accurate to at least approximately 0.25 Å or less are used. The preferred methods use nuclear magnetic resonanc ["NMR"] techniques. Particularly preferred is the rotational-echo double resonance ["REDOR"] NMR method for directly measuring ^{13}C-^{15}N internuclear distances in peptides, the most accurate current method for simply and inexpensively obtaining such distances. It is generally capable of accuracy to 0.1 Å and a span of 8 Å. In a specific embodiment, peptide binders are synthesized from amino acids labeled with ^{13}C and ^{15}N. Labeling is chosen to obtain the most useful distance data about the selected candidate pharmacophore structures. Either backbone nuclei, side chain nuclei, or both can be labeled. The step is detailed below. Liquid NMR techniques can also be used to indirectly determine internuclear distances in peptides, but are less preferred since they require considerable data interpretation to obtain distances of less accuracy than those obtained by use of REDOR.

Structure determination, step 5, determines a precise geometric conformation for both the candidate shared chemical structures, if possible, and the remainder of the binders. The preferred but not limiting method, consensus, configurational bias, Monte Carlo ["CCBMC"] determination, step 5, is an efficient smart Monte Carlo method uniquely able to incorporate knowledge from prior steps to obtain highly accurate physical binder structures. From library screening, step 2, it is deduced that the binders have a shared, actual pharmacophore, structure because they all bind specifically to the same target molecule (hence, a "consensus" method). It is not significant to the method if the binders come from more than one library as long as they all have a structure adaptable to representation in the consensus structure determination step (see *infra*). From distance measurements, step 4, a few strategically chosen distances are accurately known. This information is heuristically utilized along with an accurate model of the physical atomic interactions and the allowed molecular conformations.

Further, these means are particularly adapted for determining structures of molecules having limited conformational degrees of freedom at th temperature of interest and conformationally constrained by, *e.g.*, internal bonds. Potential conformations are generated and selected by smart configuration bias techniques which avoid generation of unnecessarily improbable new conformations. (Hence, a "configuration bias" method.) The technique is preferably applied herein to conformationally constrained peptides. A concerted rotation technique is combined with configurational bias conformation generation so that new conformations automatically preserve the internally linked backbone structure constraints. This technique is preferably applied to the preferred constrained peptide library, of a sequence comprising CX_6C (wherein X is any amino acid). The technique is also applicable to other constrained peptide libraries, to peptoid libraries, and to any more general organic diversity libraries that meet certain geometric limitations (*i.e.*, that have structures adaptable to representation in the consensus structure determination step (see *infra*)).

The methods of the invention are not limited to the use of CCBMC for determining a consensus pharmacophore structure. Alternative embodiments of this invention may use alternative structure determination methods to determine a consensus pharmacophore structure. For example, a simple yet expensive method is to make exhaustive REDOR NMR measurements characterizing the candidate pharmacophore in each binder and then average these measurements. A somewhat less expensive method is to use a conventional Monte Carlo molecular structure determination method to limit somewhat the number of REDOR NMR measurements required to characterize the candidate pharmacophore. Conventional Monte Carlo methods, being unable to directly make use of partial distance measurements or consensus binding information, are less efficient than the CCBMC method and require more distance measurements. Further, other known techniques of molecular structure determination, for example folding rules or molecular dynamics, can be used in place of conventional Monte Carlo.

The success of the structure determination is tested, step 6, against various convergence and success criteria. Consistency tests, step 6, are applied to the resulting structure to determine whether the candidate pharmacophore previously selected is the actual pharmacophore. One set of tests checks predicted distances against new distance measurements or against previous measurements temporarily not used as structure constraints. A second set of tests checks heuristically whether the candidate pharmacophore exhibits the expected low energy consensus structure. The test are described further below. If a shared structure is found, the candidate pharmacophore must be the actual pharmacophore. If not, another candidate pharmacophore and another shared structure is determined, if possible. An actual pharmacophore exists and will eventually be found and accurately structured.

Upon passing these tests, the methods of the invention have provided a consensus structure for the selected candidate pharmacophore, preferably accurate to at least approximately 0.25-0.50 Å, as well as structures for the remainder of the binder molecules. Lead compound selection, step 7, uses these structures to determine or select highly targeted lead compounds 8. One method of lead selection is to design new organic molecules of pharmacologic utility with the determined pharmacophore structure. Another method selects leads from databases of molecular descriptions. Conventionally known to medicinal chemists are databases of potential drug compounds indexed by their significant chemical and geometric structure (*e.g.*, the Standard Drugs File (Derwent Publications Ltd., London, England), the Bielstein database (Bielstein Information, Frankfurt, Germany or Chicago), and the Chemical Registry database (CAS, Columbus, Ohio)). The determined pharmacophore, being a chemical and geometric structure in the preferred embodiment, is used to query such a database. Search results will be those compounds with homologous chemical groups arrayed in a very closely similar geometric arrangement. These are lead compounds 8 output from this invention and input to the process of drug testing and development.

Although the preferred identity and ordering of the method steps is presented in Fig. 1, the invention is not limited to this identity and ordering. Other orderings, especially of steps 3, 4, and 5, are possible to achieve certain efficiencies. Steps can be inserted and deleted, for optimal effect. For example, an additional partial structure determination step can be inserted between existing steps 3 and 4 to provide information on how best to make the step 4 strategic measurements. As another example, in an alternative aspect, in lieu of screening one or more libraries to select binders, predetermined binders can be obtained and used (*e.g.*, binders determined by any means to be specific to the same target molecule); thus, step 2 can be omitted. In another embodiment, step 4, the measurement step, can be omitted. While all method steps in the preferred embodiment assume an aqueous environment at body temperature (37 °C), to the extent these parameters are relevant to the particular step, the invention is not limited to human environmental parameters.

Screening against a diversity library consists of selecting by assay those library members which bind specifically to the target molecule of interest. Binding specificity is preferably a binding constant of less than 1 µm (micromolar), and more preferably less than 100 nm (nanomolar). Preferably, an assay is done that detects an effect of binding of the binder to the target molecule on the target molecule's biological activity, to ensure that the binding is actually to the biological target of interest. Also, preferably, the selected binders are tested to further select those binders that bind to the target molecule competitively, to ensure that each binds to the same target in the target molecule.

The output of the screening step is a number, N, of binders select d from one or more libraries for use by the subsequent steps of the method. The binders with highest affinity are preferably selected for use by the subsequent steps. The chemical structure of each of the N binders selected for use is determined as part of the member synthesis and library screening. The primary chemical structure of the preferred constrained peptide library is specified by the amino acid sequence of the $-X_6-$ portion of the CX_6C molecule. For more general organic diversity libraries, the selection and arrangement of library building blocks in the binders must be determined.

It is a preferred aspect of this invention that the set of determined lead compounds is selective and small. Example 1 illustrates that as pharmacophore distance tolerances are relaxed, the number of compounds retrieved by drug database searches increases geometrically. As this invention determines high resolution pharmacophore geometries, it can be expected that database searches, or other methods of determining leads from pharmacophore structure, will return only a few, selective, targeted leads. Methods limiting the number of leads decrease the cost of drug development and are consequently of considerable utility to the pharmaceutical industry and medical community. The expense of developing and evaluating lead compounds for biological effect and medicinal usefulness is well known. Each lead compound must be screened for pharmacological usefulness, efficacy, and safety. Often chemical modifications are required and the process must be repeated. Finally, the required *in vivo* pharmacologic toxicity and clinical trials alone can consume years of time and millions of dollars.

Therefore, starting with a target molecule 1 having a biologically or pharmacologically interesting target, the method and apparatus of this invention determines a consensus pharmacophore structure. This consensus pharmacophore structure can then be used to determine a selective set of highly specific lead compounds 8 (Fig. 1) for rational design of drugs, *e.g.*, capable of acting as ligand-mimics (agonists or antagonists) for the particular target molecule.

In the following discussion and examples, each of these steps will be more fully described.

5.1. <u>SELECTION OF A TARGET MOLECULE</u>

The target molecule is any one or more molecules containing a target or putative target of interest. The target is a binding interaction region. The target can be in a single molecule or can be a product of a molecular complex. The target can be a continuous or discontinuous binding region. The target molecule selected for use (Fig. 1, step 1) is preferably any molecule that is found *in vivo* (preferably in mammals, most preferably in humans) and that has biological activity, preferably involved or putatively involved in the onset, progression, or manifestation of a disease or disorder. The target molecule can also be a fragment or derivative of such an *in vivo* molecule, or a chemical entity that contains the same target as the *in vivo* molecule. Examples of such molecules are well known in the art. Such molecules can be of mammalian, human, viral, bacterial, or fungal origin, or from a pathogen, to give just some examples. The target molecule is preferably a protein or protein complex. The target molecules that can be used include but are not limited to receptors, ligands for receptors, antibodies or portions thereof (*e.g.*, Fab, Fab', F(ab')$_2$, constant region), proteins or fragments thereof, nucleic acids, glycoproteins, polysaccharides, antigens, epitopes, cells and cellular components, subcellular particles, carbohydrates, enzymes, enzyme substrates, oncogenes (*e.g.*, cellular, viral; oncogenes such as ras, raf, etc.), growth factors (*e.g.*, epidermal growth factor, platelet-derived growth factor, fibroblast growth factor), lectins, protein A, protein G, organic compounds, organometallic compounds, viruses, prions, viroids, lipids, fatty acids, lipopolysaccharides, peptides, cellular metabolites, steroids, vitamins, amino acids, sugars, lipoproteins, cytokines, lymphokines, hormones, T cell surface antigens (*e.g.*, CD4, CD8, T c ll antigen r ceptor), ions, organic chemical groups, viral antigens (hepatitis B virus surface or core antigens, HIV antigens (*e.g.*, gp120, gp46)), hepatitis C virus antigens, toxins (*e.g.*, bacterial toxins), cell wall components, platelet antigens (*e.g.*, gpiibiiia), cell surface proteins, cell adhesion molecules, neurotrophic factors, and neurotrophic factor receptors.

In specific embodiments, vEGF (vascular endothelial growth factor) or KDR (the receptor for vEGF) (Terman et al., 1992, Biochem. Biophys. Res. Comm. 187:1579-1586) is the target molecule. vEGF and its receptor are the major regulators of vasculogenesis and angiogenesis (Millauer et al., 1993, Cell 72:835). Inhibition of the vEGF and the concomitant inhibition of its mitogenic activity and angiogenic capacity has been shown to suppress tumor growth *in vivo* (Kendall et al., 1993, Proc. Natl. Acad. Sci. USA 90:10705-10709; Kim et al., 1993, Nature 362:841-844). Use of vEGF or KDR or portions thereof, as a target molecule is a preferred embodiment for use of the present invention to develop lead molecules as drugs in the area of cardiovascular disease or cancer.

The proteins ras and raf, or portions thereof (*e.g.*, modules -- functional portions), are also preferred target molecules, particularly in an embodiment wherein the methods of the present invention are employed to develop lead molecules for drugs that are cancer therapeutics. ras is a member of an intracellular signaling cascade that controls cell growth and differentiation (Cook and McCormick, 1994, Nature 369:361-362). ras functions in signal transduction by specifically recognizing the protein raf and bringing it to the cell membrane (Hall, 1994, Science 264:1413-1414; Vojtek et al., 1993, Cell 74:205-214). The recognition modules in both ras and raf have been determined (Zhang et al., 1993, Nature 364:308-313; Warne et al., 1993, Nature 364:352-355; and Vojtek et al., 1993, Cell 74:205-214); in a specific embodiment, such a recognition module is used as a target molecule according to the invention.

In another specific embodiment, an integrin is used as a target molecule. Such molecules are known to function in clot formation, and can be used according to the present invention to develop lead molecules for drugs in the area of cardiovascular disorders.

Target molecules for use can be obtained commercially (where the target is commercially available), or can be synthesized or purified from natural or recombinant sources. In a specific embodiment, a target molecule is prepared that has been modified to incorporate an "affinity tag," *i.e.*, a structure that specifically binds to a known binding partner, to facilitate recovery/isolation/immobilization of the target molecule. In a preferred aspect, recombinant expression methods well known in the art can be used to produce a protein target molecule as a fusion protein, incorporating a peptide affinity tag. Such affinity tags include but are not limited to epitopes of known antibodies (*e.g.*, c-myc epitope (Evan et al., 1985, Mol. Cell. Biol. 5:3610-3616)), a series (*e.g.*, 5-7) of his residues (which bind to zinc), maltose binding sequences such as pmal, etc. Tags are incorporated into protein targets at either the amino or carboxy-terminus. In another embodiment, the target is chemically attached to a tag (*e.g*, biotin (which binds to avidin, streptavidin), streptavidin), *e.g.*, by biotinylation.

The target molecule is purified by standard methods. For example, a protein target can be purified by standard methods including chromatography (*e.g.*, ion exchange, affinity, and sizing column chromatography), centrifugation, differential solubility, or by any other standard technique for the purification of proteins; in a preferred embodiment, reverse phase HPLC (high performance liquid chromatography) is employed.

Once the target molecule has been purified, it is preferably tested to ensure that it retains its biological activity (and thus retains its native conformation). Any suitable *in vitro* or *in vivo* assay can be used. In instances where the desired target molecule is a fragment or derivative of a molecule found *in vivo*, or is a chemical entity putatively containing the same target as a molecule found *in vivo*, it is highly preferred that testing be done of such desired target molecules prior to their use, so that among such desired target molecules, only those that have the same biological activity as the *in vivo* molecule or compete with a known ligand to the *in vivo* molecule, are selected for actual use as target molecules according to the invention. In the event that biological activity has been reduced or lost in a recombinant protein relative to the native form of the protein, the protein can be recombinantly expressed in a different host (*e.g.*, yeast, mammalian, or insect) and/or with a variety of tags and location of tags (on either the amino- or carboxy-terminal side), in order to attempt to achieve, or to optimize, recovery of biological activity.

5.2. DIVERSITY LIBRARIES

According to a preferred embodiment of the invention, diversity libraries are screened to select binders, which specifically bind to the target molecule. Diversity libraries are those containing a plurality of different members. Generally, the greater the number of library members and the greater the probability that all possible members are represented, the more preferred the library. In preferred embodiments, the diversity libraries have at least 10^4 members, and more preferably at least 10^6, 10^8, 10^{10}, or 10^{14}, members.

Many libraries suitable for use are known in the art and can be used. Alternatively, libraries can be constructed using standard methods. Chemical (synthetic) libraries, recombinant expression libraries, or polysome-based libraries are exemplary types of libraries that can be used.

In a preferred embodiment, the library screened is a constrained, or semirigid library (having some degree of structural rigidity). Examples of constrained libraries are described below. A linear, or nonconstrained library, is less preferred although it may be used. Additionally, one or more different libraries can be screened to select binders.

In a preferred embodiment, the library contains peptide or peptide analogs having a length in the range of 5-18 amino acids or analogs thereof in each library member.

In specific embodiments, binders are identified from a random peptide expression library or a chemically synthesized random peptide library. The term "random" peptide libraries is meant to include within its scope libraries of both partially and totally random (variant) peptides.

In one embodiment, the peptide libraries used in the present invention may be libraries that are chemically synthesized *in vitro*. Examples of such libraries are given in Fodor et al., 1991, Science 251:767-773, which describes the synthesis of a known array of short peptides on an individual microscopic slide; Houghten et al., 1991, Nature 354:84-86, which describes mixtures of free hexapeptides in which the first and second residues in each peptide were individually and specifically defined; Lam et al., 1991, Nature 354:82-84, which describes a "one bead, one peptide" approach in which a solid phase split synthesis scheme produced a library of peptides in which each bead in the collection had immobilized thereon a single, random sequence of amino acid residues; Medynski, 1994, Bio/Technology 12:709-710, which describes split synthesis and T-bag synthesis methods; and Gallop et al., 1994, J. Medicinal Chemistry 37(9):1233-1251. Simply by way of other examples, a combinatorial library may be prepared for use, according to the methods of Ohlmeyer et al., 1993, Proc. Natl. Acad. Sci. USA 90:10922-10926; Erb et al., 1994, Proc. Natl. Acad. Sci. USA 91:11422-11426; Houghten et al., 1992, Biotechniques 13:412; Jayawickreme et al., 1994, Proc. Natl. Acad. Sci. USA 91:1614-1618; or Salmon et al., 1993, Proc. Natl. Acad. Sci. USA 90:11708-11712. PCT Publication No. WO 93/20242 and Brenner and Lerner, 1992, Proc. Natl. Acad. Sci. USA 89:5381-5383 describe "encoded combinatorial chemical libraries," that contain oligonucleotide identifiers for each chemical polymer library member.

In another embodiment, biological random peptide libraries are used to identify a binder which binds to a target molecule of choice. Many suitable biological random peptide libraries are known in the art and can be used or can be constructed and used to screen for a binder that binds to a target molecule, according to standard methods commonly known in the art.

According to this approach, involving recombinant DNA techniques, peptides are expressed in biological systems as either soluble fusion proteins or viral capsid fusion proteins.

In a specific embodiment, a phage display library, in which the protein of interest is expressed as a fusion protein on the surface of a bacteriophage, is used (*see*, *e.g.*, Smith, 1985, Science 228:1315-1317). A number of peptide libraries according to this approach have used the M13 phage. Although the N-terminus of the viral capsid protein, protein III (PIII), has been shown to be necessary for viral infection, the extreme N-terminus of the mature protein does tolerate alterations such as insertions. The protein PVIII is a major M13 viral capsid protein, which can also serve as a site for expressing peptides on the surface of M13 viral particles, in the construction of phage display libraries. Other phage such as lambda have been shown also to be able to display peptides or proteins on their surface and allow selection; these vectors may also be suitable for use in production of libraries (Sternberg and Hoess, 1995, Proc. Natl. Acad. Sci. USA 92:1609-1613).

Various random peptide libraries, in which the diverse peptides are expressed as phage fusion proteins, are known in the art and can be used. Examples of such libraries are described below.

Scott and Smith, 1990, Science 249:386-390 describe construction and expression of a library of hexapeptid s on the surface of M13. The library was made by inserting a 33 base pair Bgl I digested oligonucleotide sequence into an Sfi I digested phage fd-tet, *i.e.*, fUSE5 RF. The 33 base pair fragment contains a random or "degenerate" coding sequence $(NNK)_6$ where N represents G, A, T or C and K represents G or T. Cwirla et al., 1990, Proc. Natl. Acad. Sci. USA 87: 6378-6382 also described a library of hexapeptides expressed as pIII gene fusions of M13 fd phage. PCT publication WO 91/19818 dated December 26, 1991 by Dower and Cwirla describes a library of pentameric to octameric random amino acid sequences.

Devlin et al., 1990, Science, 249:404-406, describes a peptide library of about 15 residues generated using an (NNS) coding scheme for oligonucleotide synthesis in which S is G or C.

Christian and colleagues have described a phage display library, expressing decapeptides (Christian, R.B., et al., 1992, J. Mol. Biol. 227:711-718). The DNA of the library was constructed by use of an oligonucleotide comprising the degenerate codons $[NN(G/T)]_{10}$ (SEQ ID NO:8) with a self-complementary 3' terminus. This sequence forms a hairpin which creates a self-priming replication site that was used by T4 DNA polymerase to generate the complementary strand. The double-stranded DNA was cleaved at the SfiI sites at the 5' terminus and hairpin for cloning into the fUSE5 vector described by Scott and Smith, *supra*.

Lenstra, 1992, J. Immunol. Meth. 152:149-157 describes a library that was constructed by annealing oligonucleotides of about 17 or 23 degenerate bases with an 8 nucleotide long palindromic sequence at their 3' ends. This resulted in the expression of random hexa- or octa-peptides as fusion proteins with the β-galactosidase protein in a bacterial expression vector. The DNA was then converted into a double-stranded form with Klenow DNA polymerase, blunt-end ligated into a vector, and then released as Hind III fragments. These fragments were then cloned into an expression vector at the sequence encoding the C-terminus of a truncated β-galactosidase to gen rate 10^7 recombinants.

Kay et al., 1993, Gene 128:59-65 describes a random 38 amino acid peptide phage display library.

PCT Publication No. WO 94/18318 dated August 18, 1994 describes random peptide phage display "TSAR libraries" that can be used.

Other biological peptide libraries which can be used include those described in U.S. Patent No. 5,270,170 dated December 14, 1993 and PCT Publication No. WO 91/19818 dated December 26, 1991.

In a specific embodiment, a "peptide-on-plasmid" library, containing random peptides fused to a DNA binding protein that links the peptides to the plasmids encoding them, can be used (Cull et al., 1992, Proc. Natl. Acad. Sci. USA 89:1865-1869).

Another alternative to phage display or chemically synthesized libraries is a polysome-based library, which is based on the direct *in vitro* expression of the peptides of interest by an *in vitro* translation system (in some instances, coupled to an *in vitro* transcription system). These methods rely on polysomes to translate the genomic information (in this case encoded by an mRNA molecule, in some instances made *in vitro* by transcription from synthetic DNA) (*see, e.g.*, Korman et al., 1982, Proc. Natl. Acad. Sci. USA 79:1844-1848). Such *in vitro* translation-based libraries include but are not limited to those described in PCT Publication No. WO 91/05058 dated April 18, 1991; and Mattheakis et al., 1994, Proc. Natl. Acad. Sci. USA 91:9022-9026.

Diversity library screening, step 2 of Fig. 1, determines a few, N, members (compounds) from one or more libraries and their primary sequences all of which specifically bind to target molecule 1 in a similar manner. A structured organic diversity library is a prescription for the creation of a huge number of related molecules all built from combinations of a small number of chemical building blocks. Preferred div rsity libraries for use according to the invention have m mbers whose binding to a targ t molecule is characterized by configurational entropy change that are relatively small to the binding energy. This means that library members have definite structures in the bound and, especially, the unbound states. A preferred example of a chemical diversity library for use in the invention contains short peptides with a constrained conformation. Short peptides without constrained conformations are often freely flexible in an aqueous environment and adopt no fixed unbound structure. The binding of such library members is complicated by significant configurational entropy changes. To eliminate this complication, it is preferred that all library members have a constrained structure and bind to the target molecule in a specific and identifiable manner. One method of achieving constrained conformation is to require internal linking, such as by disulfide bonds.

In one embodiment, disulfide bond formation is achieved by use of libraries that contain peptides having a pair of invariant cysteine residues, preferably positioned in the range of 2-16 residues apart, most preferably 6-8 residues apart, that cross-link in an oxidizing environment to form cystines (disulfide bonds between cysteines). An example of such libraries are those containing or expressing peptides of the form $R^1CX_nCR^2$ wherein R^1 is a sequence of 0-10 amino acids, C is cysteine, X_n is a sequence of n variant amino acids (*e.g.*, if all 20 classical amino acids are represented, X means any one of the 20 classical amino acids); n is an integer ranging from 2 to 16; and R^2 is a sequence of 0-10 amino acids. R^1 and R^2 can contain invariant or variant amino acids. Another example is such libraries are those containing or expressing peptides of the form $R^1CX_nR^2$, where R^1, X, n, and R^2 are as described above; n is preferably 8 or 9. A preferred constrained peptide library, of at least 10^6 members, consists of peptides comprising the sequence CX_6C (SEQ ID NO:1), wherein C is cysteine, X is any naturally occurring amino acid, and a disulfide bond is formed between the two cysteines. Additional invariant amino acids (*.g.*, preferably no more than 5-10 amino acids) on either the amino- or carboxy-terminus of CX_6C can be incorporated as part of the peptide in this preferred embodiment. Fig. 10 schematically illustrates such a molecule. The disulfide bridge between the two cysteines acts as a sufficient conformational constraint for the preferred practice of this invention. By way of example, the library is constructed by generating oligonucleotides with the desired degeneracy to code for the peptides and ligating them into vectors of choice. These inserted oligonucleotides are suitable for both use in *in vivo* genetic expression systems exemplified by phage display, or *in vitro* translation methods based on coupled transcription and translation from DNA of interest (see below). The creation and use of an exemplary library is described in Section 6.3 hereinbelow. The invention is easily and readily adaptable to other alternative peptide libraries which include short peptides with alternative disulfide scaffolding, for example, comprising the sequence CX_nCX_mCC with two disulfide bridges, wherein n and m are each independently an integer in the range of 2-10, and X is any amino acid. More generally, any peptide library containing members of definite conformation which bind to a target molecule in a specific and identifiable manner may be used.

Further, more general, structurally constrained, organic diversity (e.g., nonpeptide) libraries, can also be used. By way of example, a benzodiazepine library (*see e.g.*, Bunin et al., 1994, Proc. Natl. Acad. Sci. USA 91:4708-4712) may be adapted for use.

Constrained libraries that can be used are also known in the art. For example, PCT Publication No. WO 94/18318 dated August 18, 1994 describes semirigid phage display libraries, in which the plurality of expressed peptides can adopt only a single or a small number of conformations. Examples of such libraries have a pair of invariant cysteine residues positioned in or flanking random residues which, when expressed in an oxidizing environment, are most likely cross-link d by disulfid binds to form cystin s. Also disclosed are librari s having a cloverleaf structure by appropriate arrang ment of cysteine residues. Also disclosed are libraries with peptides having invariant cysteine and histidine residues positioned within the random residues, or invariant histidines alone within the random residues. TSAR-13 and TSAR-14 are exemplary semirigid libraries disclosed therein.

Other conformationally constrained libraries that can be used include but are not limited to those containing modified peptides (*e.g.*, incorporating fluorine, metals, isotopic labels, are phosphorylated, etc.), peptides containing one or more non-naturally occurring amino acids, non-peptide structures, and peptides containing a significant fraction of γ-carboxyglutamic acid.

As stated above, libraries of non-peptides, *e.g.*, peptide derivatives (for example, that contain one or more non-naturally occurring amino acids) can also be used. One example of these are peptoid libraries (Simon et al., 1992, Proc. Natl. Acad. Sci. USA 89:9367-9371). Peptoids are polymers of non-natural amino acids that have naturally occurring side chains attached not to the alpha carbon but to the backbone amino nitrogen. Since peptoids are not easily degraded by human digestive enzymes, they are advantageously more easily adaptable to drug use. Another example of a library that can be used, in which the amide functionalities in peptides have been permethylated to generate a chemically transformed combinatorial library, is described by Ostresh et al., 1994, Proc. Natl. Acad. Sci. USA 91:11138-11142).

The peptide or peptide portions of members of the libraries that can be screened according to the invention are not limited to containing the 20 naturally occurring amino acids. In particular, chemically synthesized libraries and polysome based libraries allow the use of amino acids in addition to the 20 naturally occurring amino acids (by their inclusion in the precursor pool of amino acids used in library production). In specific embodiments, the library memb rs contain one or more non-natural or non-classical amino acids or cyclic peptides. Non-classical amino acids include but are not limited to the D-isomers of the common amino acids, α-amino isobutyric acid, 4-aminobutyric acid, Abu, 2-amino butyric acid; γ-Abu, ε-Ahx, 6-amino hexanoic acid; Aib, 2-amino isobutyric acid; 3-amino propionic acid; ornithine; norleucine; norvaline, hydroxyproline, sarcosine, citrulline, cysteic acid, t-butylglycine, t-butylalanine, phenylglycine, cyclohexylalanine, ß-alanine, designer amino acids such as ß-methyl amino acids, Cα-methyl amino acids, Nα-methyl amino acids, fluoro-amino acids and amino acid analogs in general. Furthermore, the amino acid can be D (dextrorotary) or L (levorotary).

By way of example, the incorporation of non-standard or modified amino acids into libraries can be done by taking advantage of concurrent development in reassigning the genetic code (Noren et al., 1989, Science 244:182-188; Benner, 1994, Trend. BioTech. 12:158-163) and the charging of specific tRNAs with the desired amino-acid (Cornish et al., 1994, Proc. Natl. Acad. Sci. USA 91:2910-2914). See also Ibba and Hennecke, 1994, Bio/Technology 12:678-682 (particularly Table I), and references cited therein. These pre-charged tRNAs are then utilized in the *in vitro* translation system to incorporate the non-standard amino acid into the library of choice. The position of incorporation can be either random (variant) or defined (invariant). The defined case can be chosen to maximize the utility of the resulting placement of the non-natural functional group to maximize either binding properties or the ability to perform structural measurements. Similar techniques may be used to incorporate non-standard amino acids into the peptides.

In a specific embodiment, an iterative approach to library construction can be taken, as structural information on the mode of binding to a given target is obtained. For example, information from structural analysis can be used to make libraries with library members containing chemical backbones that match known chemical scaffolds, enhance solubility or membrane perm ability, r duce effect of water on structure, and incorporate other physical parameters suggested by structural analysis. Use of algorithmically optimized library inserts can be used to increase the chances of finding binders of interest (*see e.g.*, Arkin and Youvan, 1992, Bio/Technology 10:297-300).

In other embodiments, the following can be used to improve library use in both phage and bacterial systems: production of libraries in bacteria which overproduce the chaperonins GroES and GroEL (Soderlind et al., 1993, Bio/Technology 11:503-507), and production in *E. coli* strains which prevent degradation in the periplasmic space (Strauch and Beckwith, 1988, Proc. Natl. Acad. Sci. USA 85:1576-1580; Lipinska et al., 1989, J. Bacteriology 171:1574-1584). Purified cofactors such as GroES and GroEL could also be directly added to an *in vitro* expression and selection system.

5.3. SCREENING OF DIVERSITY LIBRARIES

Once a suitable diversity library has been constructed (or otherwise obtained), the library is screened to identify binders having binding affinity for the target. Screening is done by contacting the diversity library members with the target molecule under conditions conducive to binding and then identifying the member(s) which bind to the target molecule. Screening the libraries can be accomplished by any of a variety of commonly known methods. See, *e.g.*, the following references, which disclose screening of peptide libraries: Parmley and Smith, 1989, Adv. Exp. Med. Biol. 251:215-218; Scott and Smith, 1990, Science 249:386-390; Fowlkes et al., 1992; BioTechniques 13:422-427; Oldenburg et al., 1992, Proc. Natl. Acad. Sci. USA 89:5393-5397; Yu et al., 1994, Cell 76:933-945; Staudt et al., 1988, Science 241:577-580; Bock et al., 1992, Nature 355:564-566; Tuerk et al., 1992, Proc. Natl. Acad. Sci. USA 89:6988-6992; Ellington et al., 1992, Nature 355:850-852; U.S. Patent No. 5,096,815, U.S. Patent No. 5,223,409, and U.S. Patent No. 5,198,346, all to Ladner et al.; Rebar and Pabo, 1993, Science 263:671-673; and PCT Publication No. WO 94/18318. S e also the references cited in Section 5.2 hereinabove (disclosing libraries) regarding methods for screening.

Screening can be carried out by contacting the library members with an immobilized target molecule and harvesting those library members that bind to the target. Examples of such screening methods, termed "panning" techniques are described by way of example in Parmley and Smith, 1988, Gene 73:305-318; Fowlkes et al., 1992, BioTechniques 13:422-427; PCT Publication No. WO 94/18318; and in references cited hereinabove. In panning methods that can be used to screen the libraries, the target molecule can be immobilized on plates, beads, such as magnetic beads, sepharose, etc., or on beads used in columns. In particular embodiments, the immobilized target molecule has incorporated an "affinity tag," as described above, which can be used to effect immobilization by attaching the tag's binding partner to the desired solid phase.

In one embodiment, the primary method of selecting from libraries is the use of solid phase plastic affinity capture to immobilize the target molecule prior to its use in the selection (screening) process. This method can be improved upon to increase throughput, selectivity and specificity. Solid phase plastic supports can be replaced with magnetic particles. In phage-based systems, large beads can be used, but these are not believed to be suitable, due to steric hindrance, for use in bacterial systems. This steric hindrance can be avoided by using high gradient magnetic cell separation with small particles (<<0.5μm) (Miltenyi et al., 1990, Cytometry 11:231-238).

In a specific embodiment involving the use of a peptide phage display library, selection of a binder protein expressed on the surface of a bacteriophage thus selects both the binder protein and the DNA that encodes it (the DNA being within the phage particle). Following binding between the target molecule and library members, phage are releas d from a solid support on which the binder-target molecule complex is immobilized, and are amplified, *e.g.*, by infecting *E. coli* and propagating each isolated binding phage. Repeating this process of affinity capture and amplification allows those peptides which bind with the highest affinity to the target molecule to be selectively enriched from the original library.

In one particular embodiment, presented by way of example but not limitation, a phage display library can be screened as follows using magnetic beads (see PCT Publication No. WO 94/18318):

Target molecules are conjugated to magnetic beads, according to the instructions of the manufacturers. The beads are incubated with excess bovine serum albumin (BSA), to block non-specific binding. The beads are then washed with numerous cycles of suspension in phosphate buffered saline (PBS) with 0.05% Tween® 20 and recovered by drawing a strong magnet along the sides of a plastic tube. The beads are then stored under refrigeration, until use.

An aliquot of a library is mixed with a sample of resuspended beads, at 4°C for a time period in the range of 2-24 hrs. The magnetic beads are then recovered with a strong magnet and the liquid is removed by aspiration. The beads are then washed by resuspension in PBS with 0.05% Tween® 20, and then drawing the beads to the tube wall with the magnet. The contents of the tube are removed and washing is repeated 5-10 additional times. 50 mM glycine-HCl (pH 2.0), 100 μg/ml BSA solution is added to the washed beads to denature proteins and release bound phage. After a short incubation, the beads are drawn to the side of the tubes with a strong magnet, and the liquid contents are then transferred to clean tubes. 1 M Tris-HCl (pH 7.5) or 1 M NaH_2PO_4 (pH 7) is added to the tubes to neutralize the pH of th phage sample. The phage are then dilut d, *e.g.*, 10^{-3} to 10^{-6}, and aliquots plat d with *E. coli* DH5αF' cells to determine the number of plaque forming units of the sample. In certain cases, the platings are done in the presence of XGal and IPTG for color discrimination of plaques (*i.e.*, *lacZ*+ plaques are blue, *lacZ*- plaques are white). The titer of the input samples is also determined for comparison.

Alternatively, as yet another non-limiting example, screening a diversity library of phage expressing peptides can be achieved by panning using microtiter plates (see PCT Publication No. WO 94/18318) as follows:

The target molecule is diluted and a small aliquot of target molecule solution is adsorbed onto wells of microtiter plates (*e.g.* by incubation overnight at 4°C). An aliquot of BSA solution (1 mg/ml, in 100 mM $NaHCO_3$, pH 8.5) is added and the plate incubated at room temperature for 1 hr. The contents of the microtiter plate are flicked out and the wells washed carefully with PBS-0.05% Tween® 20. The plates are repeatedly washed free of unbound target molecules. A small aliquot of phage solution is introduced into each well and the wells are incubated at room temperature for 2-24 hrs. The contents of microtiter plates are flicked out and washed repeatedly. The plates are incubated with wash solution in each well for 20 minutes at room temperature to allow bound phage with rapid dissociation constants to be released. The wells are then washed five more times to remove all unbound phage.

To recover the phage bound to the wells, a pH change is used. An aliquot of 50 mM glycine-HCl (pH 2.0), 100 μg/ml BSA solution is added to the washed wells to denature proteins and release bound phage. After 10 minutes at 65°C, the contents are then transferred into clean tubes, and a small aliquot of 1 M Tris-HCl (pH 7.5) or 1M NaH_2PO_4 (pH

7) is added to neutralize the pH of the phage sample. The phage are then diluted, *e.g.*, 10^{-3} to 10^{-6} and aliquots plated with *E. coli* DH5αF' cells to determine the number of the plaque forming units of the sample. In certain cases, the platings are done in the presence of XGal and IPTG for color discrimination of plaques (*i.e.*, *lac*Z+ plaques are blue, *lac*Z- plaques are white). The titer of the input samples is also determined for comparison (dilutions are generally 10^{-6} to 10^{-9}).

By way of another example, diversity libraries expressing peptides as a surface protein of either a particle or a host cell, *e.g.*, phage or bacterial cell, can be screened by passing a solution of the library over a column of the target molecule immobilized to a solid matrix, such as sepharose, silica, etc., and recovering those particles or host cells that bind to the column after washing and elution.

In yet another embodiment, screening a library can be performed by using a method comprising a first "enrichment" step and a second filter lift step as described in PCT Publication No. WO 94/18318.

Several rounds of serial screening are preferably conducted. In a particularly preferred aspect, each round is varied slightly, *e.g.*, by changing the solid phase on which immobilization occurs, or by changing the method of immobilization on (*e.g.*, by changing the linker to) the solid phase. When using a phage display library, the recovered cells are then preferably plated at a low density to yield isolated colonies for individual analysis. By way of example, the following is done: The individual colonies are selected, grown and used to inoculate LB culture medium containing ampicillin. After overnight culture at 37°C, the cultures are then spun down by centrifugation. Individual cell aliquots are then retested for binding to the target molecule attached to the beads. Binding to other beads, having attached thereto a non-relevant molecule, can be used as a negativ control.

In a specific embodiment, different rounds of screening can respectively involve selection against targets in primarily their purified form, and then in their natural state (*e.g.*, on the surface of a mammalian cell) (*see, e.g.*, Marks et al., 1993, Bio/Technology 11:1145-1149, describing selection against cell surface blood group antigens).

In other examples, subsequent rounds of screening can involve immobilization of the target molecule by attachment at different ends (*e.g.*, amino or carboxy-terminus) of the target molecule to a solid support, or presentation of library members by attachment to or fusion at different ends of the library members.

By way of other examples of screening methods that can be used, genetic selection methods can be adapted for screening of libraries, or can be used in a recursive scheme. Thus, in a specific aspect, the invention provides screening methods in which methods allowing high throughput and diversity screening (*e.g.*, screening phage display or polysome libraries against a ligand) are utilized in initial rounds, with subsequent rounds employing a genetic selection technique, in which the presence of a binder of appropriate specificity increases the activity of or activation of a transcriptional promoter or origin of replication. Genetic selection techniques that can be adapted for use (*e.g.*, by inserting random oligonucleotides in the test plasmid) include the two-hybrid system for selecting interacting proteins in yeast, replicative based systems in mammalian cells, and others (*see, e.g.*, Fields & Song, 1989, Nature 340:246-246; Chien et al., 1991, Proc. Natl. Acad. Sci. USA 88:9578-9582; Vasavada et al., 1991, Proc. Natl. Acad. Sci. USA 88:10686-10690). Thus, in a specific embodiment, compounds are produced as fusion proteins, and contacted with a different fusion protein comprising a target fused to another molecule, in which specific binding of the fusion proteins to each other results in an increase in activity or activation of a transcriptional promoter or an origin of replication. In a specific embodiment, a genetic selection method is used in a later round of screening to either select directly for a library member that binds to a target molecule, or to select a library member that competitively inhibits binding of a ligand to the target molecule.

Several exemplary methods for screening a phage/phagemid library are presented by way of example in Section 6.4 hereinbelow. An exemplary method for screening a polysome-based library is presented in Section 6.3.3 hereinbelow.

Once binders are selected from a diversity library which bind to a target molecule of interest, additional assays are preferably, although optionally, performed, including but not limited to those described below. Thus, *in vivo* or *in vitro* assays can be performed to test whether binding of a binder to the target molecule affects the target molecule's biological activity; binders that exert such an effect are preferred for use in subsequent steps of the invention. Alternatively, or in addition, competitive binding assays can be carried out to test whether the binder competes with other binders or with a natural ligand of the target molecule, for binding to the target molecule; binders that compete with each other, and that compete with the natural ligand, are preferably selected for use in subsequent steps of the invention. Alternatively, or in addition to the above assays, the binding affinity of binders for the target molecule is determined, by standard methods, or by way of example, as described in Section 6.5 *infra*. Binders of the highest affinity are preferred for use in subsequent steps of the invention.

5.4. DETERMINING THE SEQUENCE OR CHEMICAL FORMULA OF BINDERS

Many of the references cited in Section 5.2 and 5.3 hereinabove, which disclose library construction and/or screening, also disclose methods that can be used to determine the sequence or chemical formula of binders isolated from such libraries. By way of exampl , a nucleic acid which xpresses a binder can be identifi d and recovered from a peptide expression library or from a polysome-based library, and then sequenced to determine its nucleotide sequence and hence the deduced amino acid sequence that mediates binding. (In an instance wherein the sequence of an RNA is desired, cDNA is preferably made and sequenced.) Alternatively, the amino acid sequence of a binder can be determined by direct determination of the amino acid sequence of a peptide selected from a peptide library containing chemically synthesized peptides. In a less preferred aspect, direct amino acid sequencing of a binder selected from a peptide expression library can also be performed.

Nucleotide sequence analysis can be carried out by any method known in the art, including but not limited to the method of Maxam and Gilbert (1980, Meth. Enzymol. 65:499-560), the Sanger dideoxy method (Sanger et al., 1977, Proc. Natl, Acad. Sci. U.S.A. 74:5463), the use of T7 DNA polymerase (Tabor and Richardson, U.S. Patent No. 4,795,699; Sequenase™, U.S. Biochemical Corp.), or Taq polymerase, or use of an automated DNA sequenator (*e.g.*, Applied Biosystems, Foster City, CA).

Direct determination of the chemical formulas of non-peptide or peptide binders can be carried out by methods well known in the art, including but not limited to mass spectrometry, NMR, infrared analysis, etc.

In preferred aspects involving certain types of libraries well known in the art, sequencing or the use of known analytic techniques for chemical formula determination will not be necessary. In some such libraries, the identity and composition of each member of the library is uniquely specified by a label or "tag" which is physically associated with it and hence the compositions of those members that bind to a given target are specified directly (*see, e.g.*, Ohlmeyer et al., 1993, Proc. Natl. Acad. Sci. USA 90:10922-10926; Brenner et al., 1992, Proc. Natl. Acad. Sci. USA 89:5381-5383; Lerner et al., PCT Publication No. WO 93/20242). In other examples of such libraries, the library members are creat d by step wise synthesis protocols accompanied by complex record keeping, complex mixtures are scr ened, and deconvolution methods are used to elucidate which individual members were in the sets that had binding activity, and hence which synthesis steps produced the members and the composition of individual members (*see, e.g.*, Erb et al., 1994, Proc. Natl. Acad. Sci. USA 91:11422-11426).

Step 2 of the invention provides as output N binding library members (binders) and their sequences or chemical formulas.

5.5. <u>CANDIDATE PHARMACOPHORE SELECTION</u>

The prior diversity library screening, step 2, determines a set of size N of specifically binding members from one or more diversity libraries. While the binders are preferably but not necessarily isolated from one or more diversity libraries (*e.g.*, binders need not be isolated from diversity libraries; known binders can be simply provided), the following description shall refer to the preferred embodiment wherein diversity library members are the binders. It will be apparent that the description is also readily applicable to binders that are not isolated from diversity libraries.

The pharmacophore responsible for the library member binding is preferably determined by an overall select and test method in this and subsequent steps. In general, a pharmacophore is specified by the precise electronic properties on the surface of the binder that causes binding to the surface of the target molecule. In the preferred embodiment, these properties are specified by the underlying, causative, chemical structures. Chemical structures are specified generally by groups such as $-CH_2-$, $-COOH$, and $-CONH_2$. The preferred pharmacophore representation consists of a specification of the underlying chemical groups and their geometric relations. The more precisely the geometric relations are specified, the more preferred. In preferred but not limiting aspects, the geometric relations ar precise to at least 0.50 Å, and most pref rably, at least 0.25 Å. A pharmacophore will usually comprise 2 to 4 of such groups, with 3 being typical. However, for complex protein recognition targets, a pharmacophore may comprise a greater number of groups. For example, it is possibl that the entire 6 amino acid sequence, -X_6-, may be needed for a member of the preferred CX_6C library to bind to complex targets, in which case the pharmacophore includes the entire binder.

Considering by way of example, the case of binders isolated from the preferred library, of sequence CX_6C, the chemical groups defining a peptide pharmacophore are terminal groups on amino acid side chains. Typically, therefore, a sequence of two to four contiguous amino acids will contain the pharmacophore of interest. For example, Fig. 11 illustrates an Arginine-Glycine-Aspartate sequence forming a well known platelet aggregation inhibiting pharmacophore, which is defined by the positions and orientations of the adjacent -CN_3H_4, -$C\alpha H_2$-, and -COOH groups. Pharmacophores formed by discontiguous amino acids are not likely to occur in the preferred library due to the conformational constraint on the short peptide imposed by the disulfide bridge.

The selection step determines candidate amino acid sequences in each binder that define a candidate pharmacophore by the positions of their terminal groups. Candidate selection depends substantially only on the chemical structures of the amino acid side chains and terminal groups (only very rarely on backbone groups). Geometric structure is not yet available and cannot be used for candidate selection. In the preferred embodiment, amino acids are grouped into homologous groups defined by group members having similar side chain structure and activity (see *infra*). Candidate pharmacophores are found by searching the sequences of the N binders for short sequences of homologous amino acids. This search will produce at least one candidate, because all the binders share the actual pharmacophore. Several candidates will usually be found since geometric information is ignored, and the search is thereby underdetermined.

Fig. 2A illustrates an exemplary method of performing the search for homologous sequences. Although this method is illustrated as searching for homologous contiguous sequences of length 3, it is easily adaptable to search for homologies of other lengths and also for discontiguous homologous sequences. If no candidate pharmacophores of length 3 have a consistent consensus structure, then pharmacophores of length 2, 4, or longer or discontiguous sequences must be searched and selected for test. For some complex targets, the pharmacophore may include the entire variable part of the library member. The exemplary method is a simple depth-first search for matching amino acid strings. More sophisticated string search methods are known and are equally applicable to this invention.

The method begins with the administrative steps 201 and 202 of labeling the binders with integers from 1 to N and assigning the string variable 'ABC' to the next left most sequence of three amino acids to test in binder 1. If this is the first candidate selection, 'ABC' will be at the left most position in binder 1. If prior candidates have been selected, 'ABC' will be assigned one amino acid to the right of its prior assignment. The FOR loop, formed by steps 203, 206, and 207, then selects each binder from 2 to N for scanning for a sequence homologous to 'ABC'. Step 203 does loop administration. Step 206 does the scanning. If homologous sequences are found, test 207 loops back to scan the next binder. If homologous sequences have been found in all binders from 2 to N, the loop exits at step 204. In this case 'ABC' is a string in binder 1 which is homologous to other strings in all remaining binders and is thus a candidate pharmacophore. The method exits at 205 for this candidate to be structured and tested for whether it is the actual pharmacophore. If a binder does not have a sequence homologous to 'ABC', then this string is not a candidate. In this case, test 208 determines if 'ABC' is at the right end of binder 1. If so, there are no more homologies to t st for and the method exits at 209. If not, then 'ABC' is advanced one amino acid to the right 210 and the scan of all binders is repeated beginning at 203.

Fig. 2B illustrates how string variable 'ABC' is scanned across binder 1, represented schematically by 220. First, 'ABC' is assigned to $X_1X_2X_3$ at 221, then to $X_2X_3X_4$ at 222, to $X_3X_4X_5$ at 223, and finally to $X_4X_5X_6$ at 224.

Given an assignment to 'ABC', step 206 scans each other binder, for example binder K with K>1, for homologous sequences. This is simply done by comparing all contiguous substrings of binder K with 'ABC' to determine if they are homologous. They are homologous if corresponding amino acids in the substring and 'ABC' are homologous. In turn, two amino acids are homologous if they satisfy established homology rules. Each homologous sequence found in binder K defines a separate candidate pharmacophore, if sequences homologous to 'ABC' are found in all other binders.

In a case where discontiguous homologous sequences are sought, 'ABC' is assigned to amino acids in discontiguous positions in binder 1 and then compared for homologies to amino acids in the same relative positions throughout the other binders.

Various rules of amino acid homology may be used in this invention. In the preferred embodiment, amino acids are homologous if they are found in the same class of amino acids, based on side chain activity (see Lehninger, Principles of Biochemistry, (1982), chap. 5). Preferred homologous groups of amino acids are as follows. The nonpolar (hydrophobic) amino acids include alanine, leucine, isoleucine, valine, proline, phenylalanine, tryptophan and methionine. The polar neutral amino acids include glycine, serine, threonine, cysteine, tyrosine, asparagine, and glutamine. The positively charged (basic) amino acids include arginine, lysine and histidine. The negatively charged (acidic) amino acids include aspartic acid and glutamic acid. The foregoing classes may be modified by those skilled in chemical arts to create finer classifications. For example, phenylalanine and tryptophan could be placed in a separate aromatic nonpolar group. Further, homology rules could depend on amino acid sequence, such as by dividing contiguous doublets or triplets of amino acids into homology groups.

The invention is not limited to the above-described exemplary method of selecting candidate pharmacophores. Any automatic method of selecting candidates that depends only on chemical structure of binder library members, preferably expressed in terms of building block composition and sequence, can be used. For example, in the case of the preferred CX_6C library, candidates could be selected by a clustering analysis performed on the entire amino acid string in a multi-dimensional space.

This above method of selecting candidate pharmacophores is not limited to the preferred CX_6C diversity library. For example, this method is immediately applicable to any diversity library having members comprising building blocks linked by a linear backbone by simply specifying rules of homology appropriate for the building blocks. These homology rules would group building blocks presenting similar structure and reactivity to targets. This method then selects candidates comprising sequences of homologous building blocks present on all the binding library members. If the library members do not have a linear backbone, a related candidate selection method can be used. In this case, the search for homologous building blocks would need to be confined to adjacent building blocks. Adjacent building blocks in this case are those building blocks brought physically close by whatever chemical structures form the library members (instead of simply being linearly adjacent on a backbone). An adjacency determination would be specific to the particular chemical structure and would be algorithmicly specified. In addition appropriate rules of homology would be specified. The method would then select candidates comprising groups of adjacent, homologous building blocks, a group being present on each binding library member.

The above-described step is the selection step of the overall select and test method. Distance measurements and Monte Carlo structuring, steps 4 and 5, determine a consensus pharmacophore structure for the candidate, if possible. If a consensus is found, the candidate is the actual pharmacophore. If a consensus is not found, this selection step must be revisited, and a new candidate selected for test.

5.6. INTRAMOLECULAR DISTANCE MEASUREMENTS

Having obtained N binders, their chemical building block structures (chemical formula or primary sequence), and the identification of a candidate pharmacophore in each binder, steps 4 and 5 of the method of this invention cooperatively determine a precise spatial structure for the candidate pharmacophore (if it exists; if not, a new candidate pharmacophore is selected.) In the preferred (but not limiting) embodiment of this invention, N members of the CX_6C library that specifically bind to the protein target of interest have been screened; their sequences determined; and a candidate pharmacophore consisting of homologous triplets (more generally from 2 to 6 mers) of amino acids has been determined in each binder.

Step 4 measures one or more strategic distances, preferably no more than 10-20, *e.g.*, 1-10 or, more preferably, 1-5 interatomic distances are measured. The remainder of the structure is determined in subsequent steps, other than by direct measurement. The interatomic distances measured in step 4 are preferably with an accuracy of at least 2 Å, more preferably at least 1 Å or 0.5 Å or 0.25 Å, and most preferably at least 0.05 Å. Thus, in a preferred but not limiting embodiment, distances in the pharmacophore are specified to at least approximately 0.25 Å. Step 5, using the CCMBC computational method, then completes determination of the pharmacophore structure at a high resolution and the structures of the rest of the binder molecules with a secondary resolution. Having a high resolution structure for the pharmacophore of interest is orders of magnitude more useful than having a low resolution structure for an entire binder. Consequently, steps 4 and 5 focus resources on the former problem.

A distance measurement method is preferred for use if it meets certain conditions, as follows. First, accuracy of distance measurements is preferably better than at least 0.25 Å for distances on the order of those between amino acids in a peptide. Second, measurement conditions preferably approximate target binding conditions, *i.e.*, are approximately physiologic. For example, crystallization, which may induce conformational changes, is preferably avoided. Also, the employed measurement methods preferably allow one binder sample to be measured when dry, when hydrated and when bound to the target molecule of interest, thereby observing the effects of water and conformational changes on binding. Third, the measurement method is preferably quick and inexpensive.

Important advantages are conveyed by these certain conditions. First, as the method of the invention determines high resolution pharmacophore structures, use of distances less accurate than the intended results would almost certainly result in decreased resolution. Second, as the CCMBC structure determination method approximates the structural effects of hydration and target binding, use of accurate distances including the physical effects of hydration or binding helps increase the resolution of the computational results. These distances as used in the CCMBC method pull the binder structures towards a more accurate representation both of the bound, hydrated pharmacophore and also of the remainder of the binder molecule without a computationally burdensome inclusion of water molecules and without knowledge of the target molecule's structure.

REDOR NMR is the preferred method of distance determination. REDOR is a solid phase NMR technique which directly measures the inter-nuclear dipole-dipole interaction strength b tween two spin ½ nuclear speci s, denoted D_{AB} wh re

A and B are the two nuclear species measured. The inter-nuclear distance between A and B is simply determined from D_{AB} by the following equation:

$$D_{AB} = \frac{h\gamma_A\gamma_B}{2\pi R_{AB}^3} \quad (1)$$

where R_{AB} is the inter-nuclear distance, h is Planck's constant, and γ_A, and γ_B are the respective gyromagnetic ratios of nuclei A and B. REDOR is typically accurate to less than 0.05 Å and can generally measure distances up to about 8 Å.

Any two nuclear species observable and resolvable by NMR methods and, preferably, adaptable to chemical inclusion in the diversity library members of interest, may be the basis of REDOR measurements. Although the subsequent description is often directed to distance determinations between ^{13}C and ^{15}N nuclei in members of a preferred library comprising the sequence CX_6C, this invention is not so limited. One skilled in the art can readily adapt the method for use in making measurements of other types of molecules (*e.g.*, peptides and nonpeptides); additionally, other nuclear species may be used. Other common spin ½ species that can be used include but are not limited to ^{31}P and the halogen ^{19}F.

General references on NMR techniques are Slichter, Principles of Magnetic Resonance, Berlin, Springer-Verlag, (1989) and Mehring, High Resolution NMR in Solids, Berlin, Springer-Verlag (1983). REDOR references include Gullion et al., Rotational-echo double-resonance NMR, J. Magn. Res. 81:196-200 (1989); Pan et al., Determination of C-N internuclear distance by rotational-echo double-resonance NMR of solids, J. Magn. Res. 90:330-40 (1990); Garbow et al., Determination of the molecular conformation of melanostatin using 13C, 15N-REDOR NMR spectroscopy, J. Am. Chem. Soc. 115:238-44 (1993), all of which are incorporated herein by reference.

Other solid phase NMR techniques are applicable but less preferred. These include but are not limited to those disclosed in Kolbert et al., Measurement of internuclear distances by switched angle spinning, J. Physical Chemistry 98:7936 et seq. (1994), and in Raleigh et al., Rotational Resonance NMR, Chemical Physics Letters 146:71 (1988). These techniques measure homonuclear distances only to 0.5 Å accuracy and are less accurate than REDOR. Liquid phase NMR techniques of NOE (nuclear overhausser) and COESY (correlation enhanced spectroscopy) can also be used but are less preferred. They require complex interpretation to obtain comparable distance accuracy greater than 0.5 Å in small molecules with complete rotational freedom.

X-ray crystallography can also be used, although it is much less preferred, since crystallization may induce conformational changes in the binder, and since binding to the target molecule may be necessary for crystallization.

In the case of REDOR measurements of the heteronuclear distances between ^{13}C and ^{15}N, ^{13}C and ^{15}N are introduced ("labeled") at the positions between which a distance measurement is needed. The preferred embodiment of the invention measures the ^{15}N NMR resonance. Since nearly all the ^{15}N signal will originate with nuclear labels, very little background signal due to natural abundance nuclei need be accounted for. Alternatively, the ^{13}C resonance may be measured, in which case the natural abundance background is subtracted from the measurements.

Since REDOR depends on observing the internuclear dipole-dipole interaction, the binder being measured should be substantially stationary on the time scale of the NMR signal. The measurement system preferably ensures this condition. The substrate holding the binder to be measured can be chosen so as to restrain binder motion, or the measured sample may be cooled to restrain motion, or, alternatively, the binder may be bound to its target molecule in order to restrain its motion.

Further details of the REDOR distance measurements will make reference to Fig. 3. This illustrates the measurement method for one labeling of one binder, which is repeated if the binder requires multiple labelings and also is repeated for each binder. Subsequent description will focus on only one binder.

Step 41 chooses a binder labeling. Labeling is preferably done to obtain the most information about the pharmacophore consistent with chemical labeling opportunities and available labeled amino acids. Backbone labeling, for example, labels the amide N of one amino acid and one of the backbone C's of a next adjacent or more distant amino acid. Backbone labeling is typically done in the backbone in the vicinity of the candidate pharmacophore. It might also be done away from a candidate pharmacophore to confirm a previously determined structure as described for step 6. Side chain labeling strategies vary with the chemical opportunities offered by the candidate pharmacophore. If a terminal N is available, an adjacent side chain or backbone C can be labeled. If not, the side chain C and backbone amino N can be labeled. Side chain labeling is preferably on side chains in the candidate pharmacophore. Preferred labeling in the candidate pharmacophore is either a backbone amino N and a nearby backbone C or a side chain C or, if available, a side chain amino N and an adjacent or nearby side chain C.

In an alternative embodiment, to get the most structural information on the binders, these labelings are designed to select the actual major conformation from known possible conformations. For example, if it is known from preliminary determinations that a binder may exist in one of a few, e.g. two, major backbone or side chain folding patterns, the labelings are chosen to distinguish these conformations. Nuclear pairs labeled for measurement are preferably those that have significantly different distances in the possible conformations.

Multiple labeling of one binder to determine multiple distances at once is possible, for xample, by including one

^{13}C and several ^{15}N nuclei, or vice versa, in one labeled molecule. Multiple labeling is limited, however, as is obvious to one skilled in the NMR arts, by chemical shifts of the various nuclear resonances. REDOR measurement of multiple ^{15}N-^{13}C distances requires that each spectroscopically observed ^{15}N or ^{13}C resonance have a distinguishable chemical shift. If these conditions are not met, several separately labeled versions of the binder are prepared and measured, one for each internuclear distance sought.

Step 42 synthesizes the labeled binder after a labeling has been determined by applying these preferences and rules. In an embodiment wherein the binder is a peptide, variously labeled ^{13}C or ^{15}N labeled amino acid reagents for the synthesis of the labeled binder are widely available from commercial sources. A preferred supplier is Isotec Inc. (Miamisburg, OH). Other commercial sources include MSD Isotopes (Montreal, Canada) and Sigma Chemical Co. (St. Louis, MO). Step 42 has three substeps: linear peptide synthesis 43, cyclization 44 (by forming the disulfide bond), and deprotection of the side groups 45. Synthesis and side chain deprotection are performed by solid phase peptide synthesis using standard Boc (tert-butoxycarbonyl) and Fmoc (9-fluorenylmethyloxycarbonyl) chemistry. Exemplary references for this method are Merrifield, J. Amer. Chem. Soc., vol 85, pp 2149 et seq. (1963); Caprino et al., J. Amer. Chem. Soc. (1970); and Stewart et al., *Solid Phase Peptide Synthesis*, Berlin, Springer-Verlag (1984), which are herein incorporated by reference. Cyclization is by conventional mild oxidation, well known in the chemical arts. The method of these steps is detailed in Example 2 *supra*.

To obtain accurate REDOR NMR measurements, the binder sample is preferably highly purified. Accordingly, it is preferable that the sample be at least 90% pure (but not necessary if spurious NMR signals can be discriminated), and even more preferable that the sample be at least 95% pure. Such pure samples can be obtained as follows. In a first synthesis method, the binder peptide is synthesized directly on the substrate to be used in the subsequent NMR measurements. In this case particular care is preferably taken with the standard solid phase synthesis steps of Example 2. By way of example, synthesis reagents should be pure, adequate time should be allowed for diffusion of reagents and solvents throughout the interstices of the substrate resin, and between steps, prior reagents should be thoroughly washed from the resin before new reagents applied. That the purity, reaction time, and washings are adequate is gauged by subsequent analysis. An aliquot of the resulting peptide-resin is taken, the peptide is cleaved (Example 2) and its purity analyzed by mass spectroscopy or high performance liquid chromatography (HPLC).

In a second synthesis method, the peptide can be synthesized on any convenient solid phase substrate in a standard manner and then cleaved from the substrate. The peptide is purified by standard methods (e.g., HPLC) and then attached to the NMR measurement substrate. The attachment can be done by any methods known in the art, preferably at either the amino- or carboxy-terminus, e.g., by condensation of the free carboxy terminal group on the peptide with an amino labeled resin, with the attachment step preceding deprotection of any side chain carboxy groups on the peptide; by use of heterofunctional linker groups, etc.

Great care is preferably exercised in forming the binder-substrate used for the REDOR NMR measurements. This invention is also directed to binder-substrates suitable to precise REDOR NMR measurements in the following environmental conditions: dry unbound, hydrated unbound, and bound to its molecular target molecule (e.g., in lyophilized or hydrated forms).

For any binder and any NMR measurement substrate utilized, the substrate should restrain the attached binder sufficiently so that binder motion will not average out the dipole-dipole interactions necessary for the REDOR measurement. Generally, this requires that the frequency of motion of the binder be less than the frequency of the dipole-dipole interaction being observed, which varies with the nuclear species being observed and the measurement distance. For ^{13}C-^{15}N observations to 2.5 Å the binder motion frequency should be less than approximately 200 Hz; for observations to 5 Å, less than approximately 30-50 Hz; and for observations beyond 5 Å, less than approximately down to 10 Hz. The more polar the substrate, such as glass beads or p-MethylBenzhydrilamine ["mBHA"] resin, the more are polar attached binders (such as are many peptides) restrained. Less polar substrates, such as polystyrene resin, provide less restraints for polar binders. In an embodiment wherein a peptide comprising the sequence CX_6C is bound to an mBHA resin with an glycine residue serving as a linker to a binding site on the resin, probably no additional steps need be taken for 2.5 Å measurements. Additional steps that can be used, if needed, to slow binder motions include cooling the measurement sample to, for example, liquid N_2 temperatures (approximately 77 °K) or binding to a large, relatively immobile target molecule.

Second, the net binder density is important and typically is adjusted. The substrate preferably has an adjustable number of binder synthesis sites or binding sites per unit of substrate surface area. Too high a binder density on the substrate surface will cause inter-molecular nuclear dipole-dipole interactions to distort the REDOR distance measurements. To obtain accurate intra-molecular distances, the peptides should be kept sufficiently far apart so that only intra-molecular nuclear dipole-dipole interactions are significant. Inter-molecular nuclear dipole-dipole interactions are preferably kept less than about 10% of the intra-molecular interaction. In the case of ^{13}C-^{15}N measurements, this criterium can be monitored by observing ^{13}C-^{13}C dipolar couplings. As the dipole interaction falls off as R^{-3}, keeping adjacent binders apart by more than approximately 2-3 times the distance to be measured is sufficient. For measurements to 5 Å, this criterion can be satisfied by keeping binders approximately 10 Å or more apart. At a 10 Å spacing interfering ^{13}C or ^{15}N signals will not exceed 2.8 hz, which is sufficient attenuation for 30 hz or greater measurements.

In an embodiment wherein the binder is a peptide comprising the sequence CX_6C, that is synthesized on an mBHA resin that is also to serve as the NMR substrate, there is an additional upper bound on the peptide density. To prevent disulfide dimer formation in more than approximately 5% of peptides, the peptides are preferably kept apart by at least their average size. Dimer formation and incorrect disulfide scaffolds result in unconstrained, flexible peptides of altered structure distorting the REDOR distance determination of the properly conformationally constrained, cyclized binder peptides. A 10 Å or more separation will meet this requirement. In this case, more than 95% of the disulfide bonds will result in intended intra-molecular constraints. This separation may be adjusted based on a determination of actual dimer formation by chromatographic (*e.g.*, HPLC) or mass spectroscopic analysis of the peptide after cleavage from the substrate (see Section 6.6, *infra*).

NMR instrumental sensitivity places a lower bound on binder density. By way of example, for an adequate observed signal to noise ratio using a preferred NMR spectrometer, no less than approximately 10^{18} observed nuclear spins should be present in a 0.1 g sample. This translates to having a binder density of no less than approximately 0.017 mmole/g (1 mmole = 10^{-3} mole). For alternative NMR spectrometers with higher field magnets (^{1}H Larmor frequency of 500 mHz), the binder density may be as low as 0.0017 mmole/g.

A third substrate condition to be considered is pore size, which is relevant when measurement of binder bound to a target molecule is desired. In a preferred method of conducting such bound measurements, the substrate must have sufficient pore size so that the target molecules can diffuse to all binders on the surface of the substrate and bind to th m. For example, folded, moderate siz d protein targets of 50 kd are typically roughly sph rical with diameters of approximately 50 Å. Preferable substrate pore sizes for use with such moderate sized protein targets are no less than 100-200 Å. Excessive pore sizes can result in a too dilute binder that decreases NMR signal intensity. The preferable pore sizes also facilitate high purity peptide synthesis directly onto substrate resins by similarly facilitating diffusion of reagents and solvents to synthesis sites. Also, binder substrate binding is preferably of such a nature that it will not be disrupted under either dry conditions, aqueous conditions, and conditions suitable to binder-target binding. Generally, adequate pore sizes are in the range of 100-500 Å, although this will vary with the size of the target molecule.

Solid phase substrates that can be used include but are not limited to mBHA resins, divinylbenzyl polystyrene resins, and glass beads. All of these substances can be manufactured to have binding sites in the range from 0 to 1.0 mmol/g. In addition, these substrates can be made so as to have the following surface areas: for mBHA about 100 m^2/g, for polystyrene from 50-100 m^2/g, and for glass from 0.1-100 m^2/g. These substrates also can be manufactured so as to have a surface binding site density in the range of from 0 to 1.0 $mmol/m^2$. More generally any microporous material with a surface density of binding sites adjustable from 0 to at least 1.0 $mmol/m^2$, and preferably with pore sizes in the preferred ranges, can be used. Suppliers of such adjustable resins include Chiron Mimotope Peptide Systems (San Diego, CA) and Nova Biochem (San Diego, CA).

Peptide binders can be synthesized directly on the surface of the substrates, by way of example as set forth in Section 6.6 *infra*, to achieve a purity of preferably at least 90%, more preferably at least 95%. In the case of a peptide comprising the sequence CX_6C, the preferred peptide spacing on the substrate is no closer than approximately 10 Å, or a peptide density of no greater than one peptide every 100 $Å^2$. Peptide synthesis on the preferred resin p-MethylBenzhydrilamine ["mBHA"] with 0.16 mmole/g of peptide binding sites, a surface of 100 m^2/g, and a preferable pore size of 100-200 Å results in a binder-substrate having such a preferable peptide surface density and suitable for accurate REDOR NMR measurements in dry, hydrated, and bound conditions. The total binder density is more than tenfold above instrumental sensitivity. The glycine linker provides a sufficient spacer from the substrate surface.

Steps 43, 44, and 45 in the preferred embodiment of the invention are carried out by one of a number of commercial peptide synthesis sources, such as Chiron Mimotope Peptide Systems (San Diego, CA) and Nova BioChem (San Diego, CA). Methods that can be used in these steps are known in the art. However, the preferred practice of these steps is detailed in the example in Section 6.6.

The invention thus provides a method of performing solid state NMR, preferably REDOR NMR, measurements of molecules on a solid phase substrate. In one embodiment, the molecule is a compound having conformational degrees of freedom at the temperature of interest that are limited to torsional rotations about bonds between otherwise rigid subunits, the torsional rotations respecting any conformational constraints. The molecule is preferably a peptide, more preferably a peptide of constrained conformation, and is most preferably a peptide having one or more cystines (*e.g.*, comprising the sequence CX_6C). In other embodiments, the molecule is a peptide analog or derivative. In a preferred embodiment, the substrate is a solid phase on which the molecule (*e.g.*, peptide) has been synthesized, with a high degree of purity. In specific embodiments, the REDOR measurements of the molecule on the substrate can be done in a dry nitrogen atmosphere, under hydrated conditions, and when the molecule is either free or bound to a target. The invention is also directed to a solid phase substrate having a surface to which is attached a population of molecules (preferably peptides, peptide derivatives, or peptide analogs), suitable for obtaining REDOR NMR measurements of the molecules. In specific embodiments, at least 90% of the population consists of a single molecule (*i.e.*, 90% purity).

In a more preferred aspect, 95% purity is present. Methods of producing such solid phase substrates, as described above, are also provided.

Step 46 REDOR spectroscopy is performed on the strategically labeled, binder peptide-resin sample. Step 46 details include final sample preparation, spectrometer parameters and tuning, and excitation pulse sequence. Sample preparation can be carried out by standard methods. The binder peptide-substrate sample is dried in N_2, and an approximately 0.1 g amount is sealed in the NMR measurement rotor. The rotor can be cooled, if necessary, to limit binder motion.

An alternative final sample preparation step is to bind the target molecule to the binder peptide-resin sample and then dry the complex in N_2. Optionally, the binder peptide can be split from the resin before binding to the target. In this alternative, the highly accurate REDOR NMR distances are of the bound binder and thus reflect any conformational changes that occur upon binding with the target.

A triple resonance, magic angle spinning ["MAS"] NMR machine is adaptable to REDOR measurements. Such machines are commercially available from Bruker (Billerica, MA), Chemmagnetics (Fort Collins, CO), and Varian (Palo Alto, CA). An exemplary machine suitable for use is in the laboratory of Prof. Zax, Cornell University (Ithaca, NY). This machine includes a 7.05 Telsa magnet from Oxford Instruments (Oxford, United Kingdom) and RF pulse excitation and receiving hardware conventional in the NMR art. An exemplary measurement rotor is a triple resonance, MAS probe from Chemmagnetics.

The exemplary magnetic field is adjusted for a ^1H Larmor frequency of 300 Mhz with, corresponding Larmor frequencies for ^{13}C and ^{15}N of 75.4 and 30.4 Mhz, respectively. An exemplary probe spin frequency (ω_r) is 4.8 kHz, with corresponding rotor period (T_r) of 0.208 msec. ^{15}N resonances are measured. Th low natural abundance of ^{15}N eliminat s the n ed for natural background corrections. Alternatively, ^{13}C

measurements can be done with conventional background corrections.

REDOR is a pulse NMR technique requiring careful excitation of appropriate ^1H, ^{13}C, and ^{15}N resonances synchronous with the MAS rotor and followed by observation of the ^{15}N free induction decay. Many alternative REDOR excitation sequences have been described in the literature, some of which are found in the references cited hereinabove. These sequences can involve multiple ^{13}C excitations per rotor period. The simple pulse sequence preferred for use in this invention requires only one ^{13}C excitation per period.

The exemplary sequence for 8 rotor periods is illustrated in Fig. 4, and is detailed herein in a manner such that those skilled in the NMR arts can program an NMR spectrometer for similar measurement. Three channels excited are the ^1H channel 50, the ^{13}C channel 51, and ^{15}N channel 52. The ^{13}C and ^{15}N RF power supplies are tuned to the resonances of the nuclei whose distance is to be measured. The ^1H channel RF power is initially tuned to the resonance of a proton coupled to the ^{15}N of interest. The time sequence, (increasing to the right) of the exciting signals (increasing vertically) in each of these channels is illustrated.

In the ^{15}N channel, an initial excitation is applied to the ^{15}N spins in either of two manners: either an initial $\pi/2$ pulse may be applied or, as illustrated and preferred, a cross polarization transfer from the protons is made. Sufficient RF intensity is applied at time 54 in both the ^1H and ^{15}N channels, 50 and 51 respectively, to achieve a Hartman-Hahn precession match at a π spin flip time of 13.2 μsec. Subsequent to the initial ^{15}N excitation, synchronous π pulses 56 are applied in phase with the MAS probe rotor for N_c rotor cycles, denoted by line 59, with sufficient RF intensity to achieve a π spin flip time of 13.2 μsec. The phase of these π pulses is varied systematically to reduce artifacts in a manner well known in the NMR arts. The preferred sequencing is detailed in Table 1.

Table 1

^{15}N π Pulse Phase Sequencing	
Number of rotor cycles between excitation and observation	Phase sequence (in processing frame)
2	YY
4	XYXY
8	XYXYYXYX

The phase sequence is expressed as the axis, in the frame processing with the ^{15}N spins, about which the π spin flip is made. This axis is systematically varied depending on the number of rotor periods intervening between the ^{15}N excitation and signal observation. The illustrated phase sequences may be varied into equivalent sequences in a conventional manner. For example, "XYXY" is equivalent to "-YX-YX". Finally, at 501 the free induction decay of the ^{15}N spins is observed and generates the time domain output signal.

In the ^{1}H channel, the preferred sequence is an initial exciting $\pi/2$ pulse 53 followed with the previously described cross polarization transfer 54 to the ^{15}N spins. The less preferred sequence omits these initial pulses in favor of a $\pi/2$ ^{15}N excitation. During the subsequent spin evolution time for N_c rotor cycles and the free induction decay time 501, a decoupling field 55 is applied to the protons. The preferred decoupling field has a 66 kHz RF intensity to achieve a ^{1}H π spin flip in 7.6 μsec.

In the ^{13}C channel, two distinct options must be measured. The first option (not illustrated) has no ^{13}C exciting pulses. The second option (illustrated) has synchronous π pulses 57 applied for N_c rotor cycles at the rotor frequency but with a fixed phase delay 58, denoted by t_1, and at sufficient signal intensity sufficient to achieve a π spin flip time of 10.6 μsec. Any value of t_1 may be used; the preferred value is 1/2 the rotor period, $T_r/2$.

Alternative REDOR pulse sequences include 2 or more ^{13}C pulses per rotor cycle.

Summarizing still with reference to Fig. 4, a REDOR measurement scan is characterized by the number of rotor cycles, N_c, of spin evolution. A complete scan comprises, first, an equilibration period, preceding the illustrated pulse sequences. Second, there is a ^{15}N excitation period comprising pulses 53 and 54. Third, there is a spin evolution period for N_c rotor cycles which has two options, both measured. Both options comprise the application of decoupling ^1H field 55 and synchronous in phase ^{15}N π pulses 56. The first option has no ^{13}C excitation; the second has synchronous phase displaced ^{13}C π pulses 57. Fourth, and finally, there is observation of free induction decay 501 of the ^{15}N spins. Fig. 4 illustrates an N_c of 8. Each scan option is repeated, and the induction decay signal accumulated, for a sufficient number of times to obtain acceptable signal to noise ratio. With the preferred practice, this has required less than approximately 5,000 scans, and typically 3000 have been sufficient.

An alternative implementation of the REDOR measurement interchanges the roles of ^{13}C and ^{15}N and measures the free induction decay of ^{13}C. Further, the invention is not limited to this described pulse sequence and is adaptable to equivalent pulse sequences yielding direct inter-nuclear dipole-dipole interaction strengths.

Following REDOR measurement step 46, is data analysis step 47. This comprises several substeps. As is conventional, the free induction decay signal is Fourier transformed from the time domain to the frequency domain. The scan option without the ^{13}C excitation produces a transformed signal with an observed ^{15}N resonance peak of magnitude S; the scan option with ^{13}C excitation produces an observed ^{15}N resonance peak of magnitude S_f. The REDOR output signal, denoted $\Delta S/S$, is conventionally formed according to the equation:

$$\frac{\Delta S}{S} = \frac{(S - S_f)}{S} \qquad (2)$$

The output signal is observed for different N_c. Preferably 0, 2, 4, and 8 rotor cycles are observed. Other preferred N_c will be apparent during the following description.

Further analysis of the REDOR output signal, ΔS/S, is made clearer by a very brief explanation of how this output signal represents the spin 1/2 dipole-dipole interaction between the ^{13}C and ^{15}N. In the spin evolution period, the ^{1}H decoupling excitation eliminates all proton effects from the ^{13}C and ^{15}N NMR spectra. Magic angle spinning, in the scan option without any ^{13}C excitation, eliminates all nuclear dipole-dipole and chemical shift anisotropy from the NMR line. Thus signal S represents an NMR resonance without any dipole interaction. However, in the second scan option, the ^{13}C π spin flip pulses reintroduce in a controlled manner the dipole-dipole interaction. This interaction causes additional dephasing, or loss of signal strength, in the observed ^{15}N signal. Thus signal S_f represents an NMR resonance with dipole interaction and the output signal ΔS/S represents the percentage strength of pure dipole-dipole interaction between the ^{13}C and ^{15}N nuclei. The exact loss of signal strength depends on the timing of the ^{13}C pulses and the number of rotor cycles for which they are applied.

In the alternative where a general phase delay, t_1, is used, the expression for the REDOR signal is derived by numerically integrating the following equations from the Pan et al. reference (1990, J. Magnetic Resonance 90:330-340):

$$S_f = 1-\frac{1}{2\pi}\int_0^{\frac{\pi}{2}}\int_0^{2\pi}\cos[T_r\omega_D'(\alpha,\beta,t_1)]\sin\beta d\beta d\alpha \qquad (3)$$

where

$$\omega_D(\alpha,\beta,t) = \pm\frac{1}{2}D_{CN}[\sin^2(\beta)\cos 2(\alpha+\omega_r t) - \sqrt{2}\sin 2\beta\cos(\alpha+\omega_r t] \tag{4}$$
$$\omega_D'(\alpha,\beta,t_1) = \frac{1}{T_r}[\int_0^{t_1}\omega_D(\alpha,\beta,t')\,dt' - \int_{t_1}^{T_r}\omega_D(\alpha,\beta,t')\,dt']$$

This integration can be done by standard numerical integration techniques such as are found in Press et al., Numerical recipes: the art of scientific computing, Cambridge, U.K., Cambridge University Press, (1986), chapter 4, which is herein incorporated by reference. Alternatively the expression can be directly evaluated from the symbolic representations by numerical tools such as Mathematica from Wolfram Research Inc. (Champaign, IL) or Mathcad from Mathsoft Inc. (Cambridge, MA). In a preferred embodiment, however, a much simpler approach is used.

In the preferred embodiment, the ^{13}C pulse phase delay is 1/2 the rotor period, T_r, and the preceding equations can be simply expressed (Mueller et al., 1995, J. Magnetic Resonance, in press):

$$\frac{\Delta S}{S} = 1 - [J_0(\sqrt{2})\lambda]^2 + 2\sum_{k=1}^{\infty}\frac{1}{16k^2-1}[J_k(\sqrt{2}\lambda)]^2 \tag{5}$$
$$\lambda = N_c T_r D_{CN}$$

where J_k is a Bessel function of the first kind. Adequate accuracy is obtained by limiting the summation of equation 5 to its first five terms. Fig. 5 is a graph of this equation. Vertical axis 61 represents $\Delta S/S$; horizontal axis 62 represents λ; and graph 63 represents equation 5.

In detail, step 47 of Fig. 3 uses equation 5 and the REDOR output signal, $\Delta S/S$, for various values of N_c to obtain a best value for D_{CN}, the dipole interaction strength. The internuclear distance is simply and directly determined from D_{CN} by equation 1. An exemplary method for finding the best value of D_{CN} is to use a least squares method. First, form the sum of the squares of the differences of the observed $\Delta S/S$ and $\Delta S/S$ computed from equation 5, which will be a function of D_{CN}, T_r, and N_c through λ. Second, find the value D_{CN} minimizing this function by searching exhaustively in sufficiently small increments over the relevant range. For example, D_{CN} can be varied by varying R in 0.01 Å increments from 0.5 to 8 Å. More efficient minimization methods as presented in Press et al. chapter 10 can also be used. Values of the Bessel functions can be simply calculated by the methods in Press et al, *supra*, § 6.4. Alternatively, this minimization and best value determination is easily performed directly from the symbolic representations with the previously cited mathematical packages.

The example in Section 6.6 provides typical results of this measurement and analysis method.

This completes the method of Fig. 3 and determines the internuclear distance between the ^{13}C and ^{15}N nuclei to which the excitation channels were tuned for the REDOR NMR measurements. If other C-N pair distances are to be determined in the labeled binder, step 46 as detailed above is repeated for the other distinct resonances. If the alternative ^{15}N resonances cannot be distinguished, separately labeled binders are prepared and measured.

5.7. <u>CONSENSUS, CONFIGURATIONAL BIAS MONTE CARLO</u>

<u>Broad overview</u>

With reference to Fig. 1, having found N specifically binding members of one or more libraries, step 2, selected a candidate pharmacophore shared by all these binders, step 3, and determined a few strategic distances in the vicinity of the candidate pharmacophore, step 4, precise pharmacophore and binder peptide structures are now determined by the preferred method, the consensus, configurational bias Monte Carlo method. Other orderings and identities of these steps are possible. For example, the binders may be predetermined thereby rendering step 2 unnecessary. Further, no strategic distance measurements may need to be made, and step 4 may be omitted. Alternatively, a partial structure determination step may be inserted before step 4 to guide selection of distances for measurement.

Pharmacophore structure determination of this invention is not limited to the CCBMC method to be described. CCMBC makes the most efficient use of heuristic consensus binding and partial distance measurement information. However, the consensus pharmacophore can be determined by methods including but not limited to use of exhaustive REDOR NMR measurements or by extensive but fewer REDOR measurements in conjunction with a conventional molecular structure determination method, such as molecular dynamics, conventional Monte Carlo, or even peptide folding rules.

In the following description, the CCBMC method is broadly overviewed; subsequently, details of important steps are described; and finally a description of the preferred computer method and apparatus for practicing the invention is given. From the description of the methods, equations, data structures, and programs provided herein, one will be able readily to translate them into implementations.

Although the following descriptions are directed to binders isolated from the preferred library of peptides comprising the sequence CX_6C (constrained by disulfide bonds), the method is applicable to more general organic diversity library members. It is immediately applicable to compounds from constrained peptide libraries with other scaffolds and also to compounds from similar peptoid libraries. It will be readily apparent that the method is applicable to any compounds whose structural region of interest exhibits conformational degrees of freedom at a temperature of interest (*e.g.*, body temperature -- 37°C) that are limited to torsional rotations of rigid molecular subunits about bonds between the subunits, in which any loops present in the structural region of interest are independently rotatable by concerted rotation (se Section 7. Appendix: Concerted Rotation). Examples of such compounds include but are not limited to peptides, peptoids, peptide derivatives, peptide analogs, etc., including members of libraries discussed in Section 5.2, *supra*.

General features of Monte Carlo simulation methods are known. A reference is Rowley, Statistical mechanics for thermophysical property calculations, Englewood Cliffs, N.J., PTR Prentice Hall (1994), especially chapters 5 and 7, which is herein incorporated by reference. The application of simple Monte Carlo to constrained peptides has conventionally been hindered by difficulty generating geometrically proper and energetically useful conformational alterations, and by the consequent wasteful and inefficient exploration of conformational space. This method overcomes these problems for constrained peptides with a novel combination of techniques. In addition, this method is uniquely able to incorporate partial information about binding affinities and distance measurements to improve determination of the pharmacophore structure, one goal of the invention.

Fig. 8 is a overview of the method. Step 91 represents the initial geometric and chemical structure of each binding peptide in computer memory. Peptide geometric structure is represented as a set of records, each record representing one rigid subunit or one atom of the peptide. The subunit records are linked together as the subunits are linked in the peptide molecule. Each rigid unit record includes fields for the composition, structure, and connectivity of the rigid unit represented. Since the rigid units only undergo torsional rotations about mutual bonds, their internal geometric structure is fixed.

If a previous run with these peptides has been done, peptide initial structure may be chosen as one of the structures generated late in that run. Such an initial structure is desirable since the effects of arbitrary initial conditions have been eliminated. Alternatively, an initial structure is generated from a prototypical backbone without side chains by adding sidechains with random torsional orientations. For members of each type of diversity library, a prototypical backbone meeting structural constraints and representing an allowed configuration for a member possessing no side chains can be defined. The prototypical backbone for the CX_6C library is generated from the CCBMC model itself as run for the linear peptide $C(gly)_6C$ (SEQ ID NO:7) using a Hamiltonian consisting only on the H_{NMR} term. The H_{NMR} term contains only terms which, in the disulfide bond backbone region $-C_1-S_1-S_2-C_2-$, limit the S_1-S_2 distance to 2.038 Å and both the C_1-S_2 and the S_1-C_2 distances to 2.883 Å. When run for a linear peptide, no Type II backbone moves are made. Only Type I backbone moves which remove and regrow randomly selected portions of the backbone are used to generate backbone alterations. The model is run with temperatures gradually decreasing from room temperature to a small temperature, approximately 1 °K. The final low temperature structure is used for the prototyptical backbone. Backbones for similar constrained peptide libraries can be constructed in similar manners.

In memory, for each peptide, a current structure is represented; the initial current structures being the just assigned initial structures. Also in memory is represented a proposed modified structure for one peptide. At step 92 the processor generates "moves" that transform the current structure of a randomly chosen peptide into a proposed modified structure. The moves mimic body temperature (37 °C) thermal agitation experienced by the binders so that their equilibrium structure may be determined.

Generation of these moves for conformationally constrained peptides is an important aspect of this method. There are two move types. Type I moves alter the conformation of the side chain of a randomly chosen amino acid of the randomly chosen peptide. The alteration is built by side chain removal followed by side chain regrowth into a new torsional conformation. During regrowth, unfavorable overlap with neighboring side chains is avoided. Type II moves alter the conformation of a limited random region of the peptide backbone of a randomly chosen binder by performing linked, or "concerted", rotations, the linking being such that only four backbone rigid units are spatially displaced. Thereby the internally bonded ring of 8 amino acids will not be disrupted. A reference describing a similar move in linear alkane molecules is Dodd et al., A concerted rotation algorithm for atomistic Monte Carlo simulation of polymer melts and glasses, Molecular Phys., vol 78, pp 961 et seq. (1991), which is herein incorporated by reference. The ratio between the Types I and II moves is an adjustable parameter with a preferred value of 4.

Another important aspect of this method is that both moves are selected in a "configurationally biased" manner. Normal Monte Carlo methods use standard Metropolis procedures, in which each proposed structure is generated randomly and independently of the current structure with an equal a *priori* probability. However, for complex molecules, it is known that this typically results in the generation of many highly improbable or energetically unlikely structures. In some situations up to 10^5 wasted moves are generated for each useful move, a very considerable waste of processor resources. In contrast, the method of this invention generates proposed structures according to an a *priori* probability depending on the current structure and the energetic cost of the new structure. This bias toward more acceptable structures of lower energy avoids generating highly improbable structures, making a very much more efficient use of processor resources. Because detailed balance must be satisfied, the acceptance probability of the configurationally biased method must include factors in addition to the usual Boltzman factor. A reference applying a similar method for simple linear alkanes is Smit et al., Computer simulations of the energetics and siting of n-alkanes in zeolites, J. Phys. Chem. vol 98, pp 8442 et seq. (1994), which is herein incorporated by reference.

At Step 93 the processor evaluates the energy, or Hamiltonian, of the proposed configuration. The Hamiltonian contains two groups of terms: conventional physical energy terms, and heuristic constraint terms. Conventional terms include the energies of rigid unit torsional rotations and of Lenard-Jones, electrostatic interactions, and H-bonding between atoms in different rigid units. Bond lengths and angles are assumed fixed at the temperature of interest and their energies constant. These conventional interactions are exclusively intramolecular; no physical intermolecular interaction effects are considered in this invention. References for the conventional energies are Weiner et al., An all atom force field for simulations of proteins and nucleic acids, J. of Computational Chem., 7:230-52 (1986); and Weiner et al., A new force field for molecular simulation of nucleic acids and proteins, J. Amer. Chem. Soc. 106:765 (1984) (herein referred to as the "AMBER references"), which are herein incorporated by reference.

Another important aspect of the Monte Carlo method of this invention is the heuristic terms: the consensus term and the measurement constraint term. They uniquely make use of partial information on the binder peptides to guide the Monte Carlo simulation. The consensus term, $H_{consensus}$, is added to the Hamiltonian to represent that all the binders do in fact bind to the same protein target in the same physical and chemical manner. Since binding occurs at the shared candidate pharmacophore in each binder, this term makes energetically unfavorable moves that cause the geometric structure in the shared pharmacophore to depart from an average, common structure. Pseudo chemical "bonds" to this average structure are added which mimic the actual physical bonding to the surface groups of the protein target. If the candidate pharmacophore is in fact the actual pharmacophore, this energy will become minimized and small in the equilibrium configuration, since there will be an actual, shared, geometric configuration. If the candidate pharmacophore is not the actual one, this term will not become minimized or small, as there is no physical reason for this region of the peptide molecules to share a common structure. This is the only Hamiltonian term which coupl s the N binders together; no physical intermolecular effects are considered. The binders are otherwise treated independently by the method.

The measurement constraint term, H_{NMR}, is added to represent the distance measurements made, which are in fact actual distances in the molecules and constrain any simulated structure. This term makes energetically unfavorable, by adding pseudo chemical bonds of the measured lengths, moves that cause the constrained internuclear distance to depart from their measured values. Of course if no partial distance measurements have been made or are otherwise available, this term may simply be omitted from the Hamiltonian without adversely affecting the practice of this step. Which measurements to make, if any, is guided by the results of the consensus structure determined. If an adequate structure can be obtained without assistance of distance measurements, none need be incorporated. If inadequate results are obtained, additional iterations of the method will need distance measurement inputs.

Step 94 tests the proposed structure against an acceptance probability, accept(curr->prop). This acceptance probability is determined by the energy of the proposed structure previously computed in step 93. If the proposed structure fails this test and is not accepted, the method progresses immediately to step 96. If the proposed structure meets the test and is accepted, the accepted proposed structure replaces and becomes the current structure. The proposed structure of this peptide is also saved (given certain other conditions detailed later) in a separate memory store of structures for later analysis. This structure store is preferably on disk.

Repeated application of the concerted rotation may lead to a slightly imperfect structure, due to numerical precision errors. In an alternative embodiment, peptide geometry would be restored to an ideal state by application of the Random Tweek algorithm after several thousand moves (Shenkin et al., 1987, Biopolymers 26:2053-85).

Step 96 tests whether enough structures of equilibrated total energy have been generated in this simulation run. The run terminates if a sufficient number have been generated. Sufficiency is determined on the basis of whether the statistical sampling errors of the average pharmacophore structure determined at step 97 is adequate (typically, less than 0.25 Å). Preferably, 25,000 equilibrated structures would be accumulated for each run. Also, preferably, three runs would be performed for a total of 75,000 saved structures.

Fig. 9 illustrates energy equilibration of an actual run. Axis 101 is the total energy of a set of peptide binders; axis 102 is the number of moves accepted. Traces 103 represent total energies of all binders from each of the three runs. Typically, run energy rapidly equilibrates within less than approximately 2000 moves in most cases. Subsequent saved structures are counted toward termination. Traces 103 display typical energy variations superimposed on a secular stability. The illustrated energy variations typically comprise several components having different variabilities. First, there is a very high frequency oscillation with a period of a few tens of moves (known as "hair"). Second, there is a low frequency oscillation with a period of several hundred to a few thousand moves and with low amplitude.

Step 97 analyzes the structure stored in memory. In the simplest preferred embodiment, the stored geometric structures for each binder are simply averaged, yielding a final structure for each binder and for the candidate pharmacophore. In another alternative, clustering software seeks clusters of similar structures for each binder. The clusters are then averaged to give a final structure for each variant structure for each binder. The variants represent alternative foldings for the binder. Exemplary clustering methods are found in Gordon et al. Fuzzy cluster analysis of molecular dynamics trajectories, Proteins: Structure, Function and Genetics 14:249-264 (1992).

Alternative post-processing can be done on the clustered structures to account for small bond angle vibrations. Such vibrations are expect d to make small perturbations to the clustered structures determined by the Monte Carlo method and can be accounted for by a brief molecular dynamics simulation. Such a simulation is fully defined by the Hamiltonian, comprising the physical and heuristic energies to be described *infra* in Eqn. 8, and by the temperature of interest. The structures observed during the simulation are averaged to determine a final more accurate equilibrium structure. A code capable of performing such a simulation is Discover® from BIOSYM (San Diego, CA). Preferably, the molecular dynamics simulation would be run for approximately 10^5 bond angle vibration periods. Since the typical bond angle vibration period is 10^{-2} ps (1 ps = 10^{-12} sec.), such a run will encompass approximately 1 ns of molecular time.

Configurational bias move generation details

One Type I or II move will, in general, alter the position of several rigid units on a side chain or along the backbone. Each altered rigid unit is sequentially considered during move generation. The Hamiltonian describing the energy of the rigid unit currently being considered in a move is divided into an internal, u^{int}, and an external, u^{ext}, part, where u^{ext} is all energy not included in u^{int}. In the preferred embodiment, u^{int} is set to 0; an alternative choice would be to include only the torsional interaction energy between this rigid unit and units to which it is currently bound. u^{int} generates a probability distribution, p^{int}, according to which is generated a set, ϕ_k, k = 1...K, of candidate torsional angles for the bond between the rigid unit being examined and rigid units already examined. u^{ext} generates another probability distribution, p^{ext}, according to which is selected one torsional angle from the prior set as the proposed new angle for the rigid unit being examined. These probabilities are defined by the equations:

$$p_i^{int}(\phi_{i,k}) \propto \exp[-\beta u_i^{int}(\phi_{i,k})]$$

$$p_i^{ext}(\phi_{i,k}) = \frac{\exp[-\beta u_i^{ext}(\phi_{i,k})]}{w_i^{ext}} \qquad (6)$$

$$w_i^{ext} = \sum_{k=1}^{K} p_i^{ext}(\phi_{i,k})$$

In this equation, "$_i$" signifies the rigid unit being considered, K is the total number of candidate torsional angles generated by p^{int}, and $\beta = 1/kT$ (k is Boltzman's constant; T the temperature, preferably 37 °C). The overall probability of generating a transition from the current to the proposed structures and accepting the proposed structure are given by the equations:

$$P(curr\text{-}prop) \propto \prod_{i=1}^{K} p_i^{int}(\phi_{i,k})\, p_i^{ext}(\phi_{i,k})$$

$$W^{new} = \prod_{i=1}^{K} w_i^{ext} \qquad (7)$$

$$accept(curr\text{-}prop) = \min(1, \frac{W^{new}}{W^{old}})$$

In this equation, M is the total number of rigid units added in the move. W^{old} is a weight for the reverse move and will be described subsequently.

Because energy is included in the generation probabilities, proposed structures are preferentially of lower energy. Since the acceptance of proposed structures depends on their energies, the acceptance of proposed structures is thereby more probable.

Peptide memory representation details

It is well known that at body temperature peptides consist of linked rigid units capable only of torsional rotational about mutual bonds whose lengths and angles are fixed. The torsional rotations respect any molecular conformational constraints. See Cantor et al., Biophysical chemistry part I the conformation of biological macromolecules, New York, W.H. Freeman and Co. (1980), which is herein incorporated by reference. Table 2 lists the rigid units encountered in the preferred embodiment of this invention utilizing libraries of conformationally constrained peptides. Table 2, where applicable, also lists dihedral bond angles between incoming and outgoing bonds to a rigid unit and the assigned unit type.

Table 2

Type	Chemical Structure	Bond angle (if applicable)
Backbone and side chain rigid units		
A	$-NH_2$	
B	$\overset{\vert}{-C\alpha H-}$	70.5°
C	-CONH-	70.5°
D	-COOH	
Side chain only rigid units		
E	$-CH_2-$	70.5°
F	$\overset{\vert}{-CH-}$	70.5°
G	-S-	70.5°
H	$-C_6H_4-$	0°
I	$-CH_3$	
J	-OH	
K	-SH	

Type	Chemical Structure	Bond angle (if applicable)
L	$-NH_2$	
M	$-C_6H_5$	
N	$-CONH_2$	
O	$-CN_3H_4$	
P	$-C_3N_2H_3$	
Q	$-C_8NH_6$	

Table 3 illustrates the decomposition of all amino acid side chains into rigid units. Glycine is a special case, without a side chain. Proline is a special case with a side chain cyclically bonded to the backbone amino N.

Table 3

Amino Acid	Rigid Units
Glycine	$-C\alpha H_2-$ (SPECIAL CASE)
Alanine	$-CH_3$
Arginine	$-CH_2-CH_2-CH_2-CN_3H_4$
Aspartate	$-CH_2-COOH$
Asparagine	$-CH_2-CONH_2$
Cysteine	$-CH_2-SH$
Glutamate	$-CH_2-CH_2-COOH$
Histidine	$-CH_2-C_3N_2H_3$
Isoleucine	$-CH(-CH_3)-CH_2-CH_3$
Leucine	$-CH_2-CH(-CH_3)_2$
Lysine	$-CH_2-CH_2-CH_2-CH_2-NH_2$
Methionine	$-CH_2-CH_2-S-CH_3$
Phenylalanine	$-CH_2-C_6H_5$
Serine	$-CH_2-OH$
Threonine	$-CH(-CH_3)-OH$
Tryptophan	$-CH_2-C_8NH_6$
Valine	$-CH(-CH_3)-CH_3$
Tyrosine	$-CH_2-C_6H_4-OH$

Fig. 10 illustrates a structurally correct but geometrically inaccurate decomposition of the peptide backbone CX_6C into rigid units (inessential hydrogens have been omitted). Rigid units are set off in boxes 121 and their types 122 are indicated. Fig 11 illustrates a structurally correct but geometrically inaccurate decomposition of the peptide backbone and side chains of -arginine-glycine-aspartate- ("RGD") into rigid units. Rigid units are set off in boxes 131 and their types 132 are indicat d.

Rigid units are represented as records in memory. The data structure for a peptide comprises records for its constituent rigid units linked together by data pointers exactly as the actual rigid units in the peptide are chemically linked. The record representing a rigid unit comprises fields for: type of the unit, pointers to chemically bonded units, all atoms of the unit and their spatial positions, atoms of the unit that are the target of the incoming and outgoing bonds, amino acid to which the unit belongs, and atomic composition of the unit.

A known, conventional representation of atoms and atomic interactions is taught by the AMBER references. Each atom is divided into a series of subtypes of specific properties. For example, for carbon there are subtypes C, C2, CA, CT, etc.; for nitrogen, there are N, N2, etc.; for oxygen, there are O, O2, etc.; and for hydrogen, there are H, H2, etc. Bonds between each pair of subtypes are separately characterized by equilibrium lengths, angles, and torsional energies. Interactions between each pair of subtype atoms are separately characterized by Lenard-Jones force parameters, hydrogen bonding force parameters, and electrostatic charges. Amino acid charge distributions are in Weiner et al., J. of Computational Chem., 7:230-52 (1986).

Thus each atom in each rigid unit is represented by an in-memory record comprising fields for: its AMBER reference subtype and any electrostatic charge. The atom's spatial position relative to its containing rigid unit, stored in that unit's record, is geometrically determined from the unit's internal chemical structure and bonds by the AMBER bond lengths and angles defined for each of these bonds. The relative spatial positions of atoms within a rigid unit are, of course, fixed, and there is no interaction energy to consider between atoms within a rigid unit.

Fig. 11 is a complete memory representation of a tripeptide sequence -RGD- (a known pharmacophore). Rigid units are set off in boxes 131 and their typ s 132 are indicated. The torsional degrees of freedom between the rigid units are indicated by angle arrows 133. AMBER atoms types are indicated as at 134. Net atomic charges are indicated only for arginine as at 135. Rigid unit records are linked into a data structure modeling the rigid unit's physical linkages. Not shown are relative atomic spatial positions represented by the atoms rectangular coordinates.

All parameters defining the AMBER atomic representations and interatomic forces can be found in Weiner et al., J. of Computational Chem., 7:230-52 (1986), and Weiner et al., J. Amer. Chem. Soc., 106:765 (1984). Conventionally, these parameters are obtained from computer readable files from commercial sources. The preferred computer readable source of these parameters is from Insight II® 2.3.5 software from BIOSYM (San Diego, CA). Other sources are Tripos (St. Louis, MO) and CHARMm (Molecular Simulations, Inc., Burlington, MA).

Interaction energy evaluation details

The form of the intramolecular energy, or Hamiltonian, evaluated at step 93, is an important element of this invention. The Hamiltonian consists of the components:

$$H_{total} = \sum_{l \in binders} H_{l,\,total} \qquad (8)$$

$$H_{l,\,total} = H_{l,\,molecular} + H_{l,\,NMR} + H_{l,\,consensus}$$

The $H_{l,molecular}$ component is determined from the Weiner et al. references, J. of Computational Chem., 7:230-52 (1986), and J. Amer. Chem. Soc., 106:765 (1984).

$$H_{l,molecular} = \sum_{\substack{n;\,i \in \\ \text{rigid unit} \\ \text{torsional} \\ \text{angles}}} \frac{V_{in}}{2}\left(\cos(n\varphi_{l,i} - \gamma_i) + 1\right) + \sum_{\substack{i<j \\ i,j \in \\ \text{atom pairs}}} \left[\frac{A_{ij}}{R_{l,ij}^{12}} - \frac{B_{ij}}{R_{l,ij}^{6}}\right] \qquad (9)$$

$$\sum_{\substack{i<j \\ i,j \in \\ \text{atom pairs}}} \left[\frac{q_i q_j}{\epsilon R_{l,ij}}\right] + \sum_{\substack{i<j \\ i,j \in \\ \text{H-bond pairs}}} \left[\frac{C_{ij}}{R_{l,ij}^{12}} - \frac{D_{ij}}{R_{l,ij}^{10}}\right]$$

Here, $\phi_{l,i}$ is the i'th torsional angle between rigid units of the l'th binder peptide, and $R_{l,ij}$ is the interatomic distance between the i'th and j'th atoms in different rigid units of the l'th binder. The first term in this equation is the torsional energy of rigid units; the second is the interatomic Lenard-Jones energy; the third is the interatomic electrostatic energy; and the fourth is the interatomic hydrogen bond energy. Rigid unit torsional rotations directly change the first term. Such rotations indirectly change all other terms as interatomic distances change.

The AMBER parameters V_{in}, A_{ij}, B_{ij}, q_i, C_{ij} and D_{ij} are obtained as stated above. The effect of water is approximated in a known manner by setting ϵ equal to $4\epsilon_0 r$, where r is distance (in Å) in the electrostatic term and ϵ_0 is the vacuum permeability.

The distance constraint term, as described, makes energetically unfavorable moves which cause those measured interatomic separations in the simulation to depart from their measured values. If no measured values are available, this term is simply omitted from the Hamiltonian. Since this is not a physical energy and in simulation equilibrium the binders should have the measured distance, it is advantageous that this term should make only a small contribution to the equilibrium energy, no more than 10% of the total energy and preferably approximately 2.5 to 5%. Further, it is advantageous that the energetic disfavor be weighted by the confidence in the measurements, so that measurements having more confidence have a greater effect.

Many forms of this energy meet these criteria. The preferred form is:

$$H_{l,NMR} = \sum_{\substack{i<j \\ i,j\in \\ \text{observed} \\ \text{distance pairs}}} \frac{(R_{l,ij} - R_{l,ij}^{(o)})^2}{2w_{l,ij}} \qquad (10)$$

where $R^{(o)}_{l,ij}$ is a measured distance in the l'th binder peptide between atomic pair ij. This makes the constraints appear as an elastic pseudo-bond with equilibrium length as measured. The $w_{l,ij}$ are weights designed to meet the above size criteria. In the preferred embodiment, they are calculated with an overall multiplicative factor limiting the contribution of $H_{l,NMR}$ to no more than approximately 5% of the total equilibrated energy. Their relative value is selected to reflect the lower reliability of longer measurements. Thus if $R^{(o)}_{l,ij}$ is between 0 and 3 Å, $w_{l,ij}$ has a relative value of 1; if the measurement is between 3 and 4.5 Å, the relative value is 2; if between 4.5 and 7 Å, the value is 3; and if the distance exceeds 7 Å, the term is dropped from the sum. Other alternative weight assignments meeting the general criteria are clearly possible.

The consensus constraint term, as described, makes energetically unfavorable moves which cause the candidate pharmacophore in each of the binders to depart from an average, shared configuration. In simulation equilibrium when the candidate is the actual pharmacophore, the binders share the pharmacophore structure and this term should be small. Since this is not a physical energy, in the case where the candidate pharmacophore is correct, this term should not be large compared to the total energy, in equilibrium no more than 10% of the total energy, and preferably approximately 5%. Further, the energetic disfavor should preferably be weighted by the affinity of each binder for the protein target, so that binders with greater affinity have a greater energetic effect.

Many forms of this energy meet these criteria. The preferred form is:

$$R^{(c)}_{ij} = \sum_{l \in binders} \frac{R_{l,ij}}{N}$$

$$H_{l,consensus} = \sum_{\substack{i<j \\ i,j \in \\ \text{pharmocophore} \\ \text{distance pairs}}} \frac{(R_{l,ij} - R^{(c)}_{ij})^2}{2w'_{l,ij}} \quad (11).$$

$R^{(c)}_{ij}$, the shared consensus structure for the candidate pharmacophore, is an average of the interatomic distances between corresponding atomic positions, ij, in the shared pharmacophore in all binders. This makes the constraints appear as a pseudo-bonds to a shared pharmacophore, which represents the binding to the protein target. The $w'_{1,ij}$ are weights designed to meet the above size criteria. In the preferred embodiment, they are calculated with an overall multiplicative factor limiting the contribution of $H_{1,consensus}$ to no more than approximately 5% of the total equilibrated energy. Their relative value is selected to reflect that binders with lower affinity are less reliable indicators of actual pharmacophore structure. Thus the relative value of the weights is proportional to the logarithm of the affinity of the corresponding binder with an affinity of 1 μmolar having a relative weight of 1. Other weight assignments meeting the general criteria are clearly possible. The heuristic $H_{consensus}$ is the only Hamiltonian term linking together the various binders.

All Hamiltonian components change only due to the dependence of the interatomic distances, $R_{1,ij}$, on the rigid unit's torsional rotation. The $R_{1,ij}$ are the well known Euclidean distances between the atomic coordinates stored in the rigid unit records. Calculation of coordinate changes due to rotation of angle ϕ about a bond with unit direction $\underline{n}$ originating at atom A with position $\underline{x}$ is well known, but will be detailed. (Throughout, symbols representing vector quantities are indicated by underlining.) First, translate from the current coordinate origin to an origin at position $\underline{x}$ by adding $\underline{x}$ to all relevant coordinate vectors. Second, apply a rotation matrix, T, to the atomic coordinate vectors. Third, translate back to the prior coordinate origin from $\underline{x}$ by subtracting $\underline{x}$ from all relevant coordinate vectors. A rotation matrix is given by:

$$T=\cos(\varphi)I+nn^T[1-\cos(\varphi)]+M\sin(\varphi)$$

$$M=\begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \qquad (12)$$

A reference for this computation is Goldstein, Classical mechanics, Massachusetts, Addison-Wesley (1981), especially chapter 4, which is herein incorporated by reference.

Type I move generation

Type I moves alter side chain structure of a randomly chosen amino acid in a randomly chosen binder. These random choices are conventionally made by a random number subroutine. The chosen side chain is "removed" from the binder peptide and "grown" back rigid unit by rigid unit. For the next, i'th, rigid unit to be added, K possible new torsional angles are generated according to p^{int}. Preferably K is from 10 to 100. One of these torsional angles is selected according to p^{ext}, and the rigid unit is added at this new angle. Determination of p^{ext} requires obtaining the normalization w_i^{ext}. At each step the u^{int} and u^{ext} used to calculate the respective probabilities include only interaction energies with rigid units present in other amino acids or already grown back. Rigid units not yet added are ignored. After all the side chain rigid units have been added back, W^{new} is computed as the product of the normalization factors.

Fig. 12 illustrates a Type I move for glutamate. At 141 the side chain has been removed. The first $-CH_2-$ unit is added back at 142 with new torsional angle ϕ_1. The generation according to p^{int} and selection according to p^{ext} of this angle ignores energy interactions with the other side chain rigid units not yet added. At 143, the next $-CH_2-$ rigid unit is added back at angle ϕ_2. Finally at 144, the last $-CO_2$ rigid unit is added at angle ϕ_2. For this last step interaction energies with all the rigid units are considered in generating and selecting the new angle.

W^{old} is the weight for the reverse move, the move from the proposed new structure to the current configuration. For this, the proposed side chain is removed and regrown in its current structure unit by unit. For the next, i'th, unit generate K-1 possible new torsional angles according to p^{int}, again ignoring interactions with units yet to be added. The K'th new angle is the current angle for that unit. The current torsional angle is selected. Although p^{ext} is not used, normalization w_i^{ext} is determined. After all units have been regrown at the current angles, W^{old} is computed as the product of the normalizations.

The acceptance probability for the proposed side chain configuration is determined from equation 7 using W^{new} and W^{old}

<u>Type II move generation</u>

Type II moves alter a limited region of the amino acid backbone beginning at a randomly chosen backbone rigid unit of a randomly chosen binder peptide in a manner consistent with conformational constraints due to internal disulfide bonds. These random choices are made similarly to those for Type I moves.

In Type II moves, side chains attached to the altered rigid units move rigidly with their backbone rigid units.

For this move, important geometric constraints must be met. In a randomly chosen binder and at a randomly chosen backbone bond between adjacent rigid units, a torsional angle rotation by ϕ_0 is made. Subsequent backbone torsional rotations are chosen so that a minimum number of rigid units undergo a spatial displacement. This constraint fixes a limited number (if any) of possible subsequent torsional angles as a function of ϕ_0 so that at most 4 rigid units are spatially displaced and rotated with at most 3 additional rigid units undergoing a rotation. This move is an important aspect of this invention and is required to maintain the conformational constraint due to the disulfide bridge. Since only 7 rigid units are spatially modified, the Type II move preserves the 8 amino acid cycle (20 rigid units), including the cystine side chain.

Fig. 13 illustrates a Type II move of a poly-glycine 7-mer. Rigid unit positions are indicated generally by black circles as at 1509 with incoming bonds generally as at 1502. A C_α rigid unit (B unit) is illustrated in box 1515, and an amide bond (C unit) in box 1516. Backbone structure 1500 in transformed into structure 1501 by the Type II move generated by an initial rotation about bond 1502. Subsequent rotations about bonds 1503, 1504, 1505, 1506, 1507, and 1508 are thereby determined so that the rigid unit 1510 and at most three subsequent units undergo only a rotation without any spatial displacement. The four rigid units between units 1509 and 1510 undergo both a spatial displacement and a rotation as structure 1500 is transformed to structure 1501. No other backbone rigid units are altered.

The derivation of these assertions, including expressions for the allowed angles, is in Section 8. Appendix: Concerted Rotation. Fig. 14 defines notation used in this Appendix: Concerted Rotation. Poly-glycine 7-mer backbone 1600 is the same as in Fig. 13. Rigid unit positions are indicated generally by black circles as at 1601 with incoming bonds generally as at 1602. The torsional rotations ϕ_0 to ϕ_6 are about bonds 1602 to 1608, respectively, between sequential, adjacent rigid units. The rigid unit position vectors $\mathbf{r}_0$ to $\mathbf{r}_6$, illustrated as vectors 1610 to 1616, respectively, define the position of these sequential rigid units with respect to a laboratory coordinate system with origin 1609. Summarizing this Appendix, the determination of the fixed torsional angles proceeds as follows. The allowed values for ϕ_1 are the roots of equation 34, which depends on the ϕ_0 driver angle and ϕ_2 through ϕ_4. But ϕ_2 through ϕ_4 can be determined in terms of ϕ_1. Two solutions for ϕ_2 are determined by quation 25 in terms of ϕ_1. Two solutions for ϕ_3 are determined by equation 29 in terms of the preceding ϕ's. Finally, a simple inversion of equation 32 determines one solution for ϕ_4 in terms of the preceding ϕ's. Having found the allowed values of ϕ_1, then equations 25, 29, and 32 determine corresponding allowed values for the other ϕ's, which in turn determine the alteration of the first four rigid units caused by the ϕ_0 initial rotation.

More precisely, final torsional angles ϕ_0 to ϕ_6 determine position vectors $\underline{r}_1$ to $\underline{r}_4$ by applying rotation matrix 18 to equations 17 to obtain new position vectors in the laboratory coordinate system, the rotation matrices of equations 16 and 18 being determined by these final torsional angles. Position vectors $\underline{r}_0$ and $\underline{r}_5$ to $\underline{r}_7$ do not change. Then rigid unit 0 is translated to position $\underline{r}_0$; aligned so that its incoming bond axis is along the direction of the outgoing bond of unit -1; and finally rigidly rotated so that the end of its outgoing bond is at position $\underline{r}_1$. Rigid unit 1 is then translated to position $\underline{r}_1$; aligned so that its incoming bond axis is along the outgoing bond of unit 0; and rigidly rotated so that the end of its outgoing bond is at position $\underline{r}_2$. Rigid units 2 to 6 are then added to the backbone in a similar fashion. In this fashion the Type II move geometry is determined. Any side chains attached to these rigid units are rigidly rotated when their parent unit is rotated.

The Type II rotation is chosen in the following manner. Using the configurational bias prescription, the Hamiltonian is divided into u^{int} and u^{ext}. u^{int} is preferably 0, or alternatively is the torsional energy associated with the rigid unit of interest, while u^{ext} includes all remaining interaction energies. In the previous manner, u^{int} determines p^{int} according to which are generated K' candidate ϕ_0 rotation angles. Preferably K' is 1. Then the geometric constraints are solved for each candidate ϕ_0. Typically, but not always, 6K', denoted K, possible backbone alterations are obtained. One of these is selected by p^{ext}, determined by:

$$p^{ext}(\phi_{0,k}) = \frac{\exp[-\beta u_c^{ext}(\phi_{i,k})]}{W^{ext}(\phi_{i,k})} \tag{13}$$

$$W^{ext}(\phi_{i,k}) = \sum_{k=1}^{K} \exp[-\beta u_0^{ext}(\phi_{i,k})]$$

u^{ext} includes all interactions not in u^{int}, that is all other backbone and side chain interactions. Because these determinations occur in torsional angle space and change the volume element in that space, the Jacobian, determined by equation 35, of the selected Type II move is also needed as a weight in the acceptance probability for detailed balance. This acceptance probability for Type II moves is:

$$accept(curr\text{-}prop) = \min[1, \frac{W^{new}J^{new}}{W^{old}J^{old}}] \tag{14}$$

The weight and Jacobian of the reverse transformation from the proposed to the current structure are also needed in the acceptance probability for Monte Carlo detailed balance. These quantities are determined as follows. Using the proposed backbone structure just selected as the basis, generate a set of K'-1 new ϕ_0 torsional angles according to p^{int} and also include the current ϕ_0 in the set. Then solve the geometric constraint to determine the permitted alterations. The current configuration, since it exists, must be among the permitted structures. From this set of permitted structures determine W^{old} per equation 13. Then select the current configuration and compute the Jacobian J^{old} per equation 35. This completes the determination or the acceptance probability.

Proline is approximated. Proline is not subject to Type I moves. However, proline is subject to normal Type II moves, with its side chain bond to the amino nitrogen broken. The side chain thus moves rigidly with its backbone rigid unit as in normal Type II move. To compensate for the broken bond approximation, the C_α-N torsional energy amplitude in the proline backbone is set at approximately 5 kcal/mole. (By contrast the torsional energy in a typical amino acid of the C_α-N bond is approximately 0.3 kcal/mole.) This invention is adaptable to other suitable approximations for proline. Alternatively, the proline side chain may be subject to alterations which preserve its cyclicity, such as for example, by an extension of the constraint scheme just described.

Program detailed description

The following describes the construction and use of a computer method and apparatus to perform the method of step 5. The listing of this code is included in a microfiche appendix to this specification. Fig. 15 is a general view of the computer system and its internal data and program structures. To the left in Fig. 15 are the principal data structures of this method. Current structures 1701 contains the current structures of the N binders represented in memory as described. Proposed structure 1702 contains working memory areas used to generate a proposed new structure for one binder peptide. Structures 1701 and 1702 would typically be stored in RAM memory of the computer system, RAM memory being memory directly accessible to processor fetches. Stored structures 1703 contain similar memory representations of all the peptide structures generated, accepted, and selected for storage. This is typically stored on permanent disk file(s).

Candidate pharmacophore structures 1704 are input to the programs from either a disk file of the display and input unit 1712. The identified candidate structures are used to determine the $w'_{1,ij}$ in Eqn 11.

Parameters 1705 comprises several parts. First, are all the AMBER atomic interaction definitions and parameters. Second, are standard representations of the amino acids including component rigid units and atomic charge assignments. Third, are parameters controlling the run.

These further comprise, by example, values for K and K', the Type I/II move branching ratio, the number of moves made in the simulations run, the simulation total energy record, etc. The parameters would typically be loaded from disk file(s) into RAM memory for manipulation during a simulation run.

Unit 1712 includes display and input devices for monitoring and control. Depicted on the display are the total number of moves made in the current run and the course of the total energy, which is similar to that illustrated in Fig. 9.

Processor 1711 is loaded with necessary programs prior to a simulation run and executes the programs to perform the simulation method. The general structure consists of main program 1706, structure modification program 1707, Type I and II move generators 1708 and 1709, and subroutines 1710. The subroutines consist of common utility subprograms, such as for performing torsional rotations about bonds and computing interaction energies by the previous methods, and conventional library subprograms, such as for performing input and output and finding random numbers. Any scientifically adequate random number generator can be used. A reference for random number generators is Press et al., Numerical recipes: the art of scientific computing, Cambridge, U.K., Cambridge University Press, (1986), chapter 7. The invention is equally adaptable to other program structures that will occur to those skilled in computer simulation arts.

The preferred embodiment of these structure is an Indigo 2 workstation from Silicon Graphics (Mountain View, CA). Alternatively, any high performance workstation, such as products of Hewlett-Packard, IBM or Sun Microsystems, could be used. Preferably the data and program structures are coded in the C computer language. Alternatively any scientifically oriented language, such as Fortran, could be used. Conventional subroutine and scientific subroutine libraries are used where appropriate.

The program components will be now described in detail with reference to Figs. 16, 17, 18, and 19. Fig. 16 illustrates main program 1706. The peptide sequences of the N binders are input at step 1801. All necessary AMBER parameters - bond lengths and angles, atomic types and charges, interaction parameters, amino acid definitions, etc. - are input at step 1802. Step 1803 creates initial structures from this input data. Rigid unit records for all rigid units are created and linked to represent peptides. The geometric structures of these peptides either are obtained from a prior run or are built by adding side chains to a prototypical backbone characteristic of the library of the binder. A prototypical backbone for the CX_6C library is found in the microfiche appendix heading CX6C.CAR. The initial binder structures are stored in the current structure data areas in preparation for the beginning the main steps of the method.

Step 1804 begins the main loop of the simulation with the generation of a proposed modified structure for one of the binder peptides by structure modification program 1707. As part of proposed structure generation, an acceptance probability, accept(curr->prop) is determined as previously described. The proposed structure will be accepted at 1805 based on this probability. For example, a random number between 0 and 1 is generated, and the proposed structure accepted if the random number is less than the acceptance probability. If the proposed structure is accepted, then it is tested for sufficient distinctiveness at step 1806. This test is met if at least one atomic position in the proposed structure differs from the corresponding position in the current structure by at least approximately 0.2 Å. If the proposed structure is distinct, it is stored at 1807 in the structure store for later analysis. Whether distinct or not, the accepted proposed structure for the peptide replaces the corresponding current structure at step 1808.

The simulation is tested for completion at step 1809. Completion can be controlled by the operator at station 1712 depending on display of run progress results. Alternatively, termination can be mechanically controlled. After completing a certain number of total moves after run energy equilibration, the moves being split between Types I and II according to the specified branching ratio, the run is terminated. The preferred number of total moves is 25,000, and the preferred Type I/II branching ratio is 4. Thus it is preferred to have 20,000 Type I and 5,000 Type II moves after equilibration per simulation run.

At step 1810, the stored structures are analyzed to determine both the consensus pharmacophore structure and the structures of the remainder of the binders. In the preferred embodiment, atomic positions in the equilibrated stored structures for each peptide are averaged to obtain the predicted geometric structure. The shared pharmacophore structure is obtained from the predicted structure of each peptide, again by averaging the shared position information for all peptides. Alternatively, before structure averaging, the structures generated for each binder can be clustered into similar groups and the clusters for each peptide separately averaged. The clusters would represent alternative peptide folding patterns. It is anticipated that because preferred binders are short peptides constrained by disulfide bridges, any alternative foldings identified will be structurally similar. The clustering can be done by the exemplary methods found in the previously referenced article Gordon et al. <u>Fuzzy cluster analysis of molecular dynamics trajectories</u>. Proteins: Structure, Function, and Genetics 14:249-264 (1992). For all analysis methods, the choice of the preferred number of stored moves is adjusted to achieve adequate estimated statistical position errors. Further, preferably, the results of three runs are combined to achieve increased statistical confidence.

Other information is also output. Particularly important is the course of the total energy for each peptide and for all the peptides, and the intra-molecular, consensus, and constraint components of the energies. These energy components are used in determining whether a consensus pharmacophore has been found. As previously described, this is preferably done by insuring that $H_{consensus}$ is small compared to the total energy and is minimized by a particular candidate pharmacophore. Also H_{NMR} must be relatively small.

Finally at 1811, all results are output in a form usable for the subsequent steps 6 and 7 of Fig. 1. For example, this may be a particular file format suitable for subsequent lead compound search by a database query.

Turning now to Fig. 17, structure modification program 1707 will be described. This is invoked from the main program at 1804. Upon entry, this program randomly picks one of the binder peptides at 1901 for which to generate a proposed structure and also picks which type of move to use at 1902. This latter random choice is made according to an adjustable Type I/II branching ratio (preferably 4). For a Type I move, step 1903 picks a random amino acid side chain of the selected peptide, and step 1904 invokes the Type I move program. (Proline has no Type I moves.) For a Type II move, step 1905 picks a random backbone bond between rigid units to rotate and also a random direction from the picked bond along which backbone rigid unit structure will be altered. Step 1906 invokes the Type II move program.

Figs. 18A and 18B illustrate the Type I move generator 1708, which is defined by equations 6 and 7. With reference first to Fig. 18A, the proposed structure of the selected peptide is created from its current structure by removing the selected side chain. All intra-molecular interactions are subsequently determined with respect to the proposed structure absent side chain rigid units not yet regrown. K candidate new torsional angles for the next, i'th, rigid unit to add are generated by p_i^{int} at 2002. Preferably K is between 10 and 100. Generation of these angles uses the conventional rejection method referenced in Press et al. at § 7.3. The weight w_i^{ext} and p_i^{ext} are determined for each of these candidate angles. This requires the rigid unit to be added to be rotated to the candidate angle using the previous rotation method. Candidate interaction energy is determined from candidate interatomic distances resulting from the candidate rotation. One of the candidate angles is probabilisticly selected at 2003 and the rigid unit added back at this torsional angle at 2004. If there are more units to add, which is tested at 2005, these steps are repeated. If not, the acceptance weight W^{new} is determined as the product of the w_i^{ext} at 2006. Lastly the old weight is determined at 2007. From the weights the move acceptance probability is found for use at 1805.

Fig. 18B details the determination 2007 of W^{old}, the weight for the reverse move from the proposed to the current side chain structure. Temporarily the proposed structure is used as a basis for energy determination at 2008, and then the current structure is restored at 2016, when this process is finished. The proposed side chain is removed at 2009 for regrowth rigid unit by rigid unit as in Fig. 18A. For the next, i'th, rigid unit to be added back, K-1 candidate angles are generated according to p_i^{int} at 2010 with the current value of that angle for the K-th candidate at 2011. As previously, the weight w_i^{ext} is determined for these candidate angles at 2012. The rigid unit is added back at the current, K-th, angle at 2013. If there are more units to add, tested at 2014, these steps are repeated. If not, the acceptance weight W^{old} is determined as the product of the w_i^{ext} at 2006.

Figs. 19A and 19B illustrate Type II move generator 1709, which is defined by equation 13 and 14 and the concerted rotation geometric constraints. With reference to Fig. 19A, K' candidate new torsional angles for the selected backbone bond are generated by p^{int} using the rejection method. Preferably K' is 1. Torsional rotations about adjacent backbone bonds, in the selected direction along the backbone, permitted by the concerted rotation constraints are determined from the roots of equation 34 at 2102. Equation 34 depends on intermediate variables obtained from equations 25, 29, and 32 and determined in that order. The roots are simply found by searching the interval $[-\pi,\pi]$ in 0.04°

increments. When a root is located in a 0.04° segment, it is refined with the bisection method referenced in Press et al. at § 9.1. It is expected on the average that six K' solutions will be found. If no roots are found at 2103, the candidate rotation is impossible and this move is skipped. If solutions exist, next, at 2104, p^{ext} and W^{new} are determined. Using the described rotation method, the backbone rigid units are rotated (with consequent spatial displacement of 4 units) to a candidate torsional angle solution about their mutual bonds. Additionally, any side chains attached to backbone rigid units are rigidly rotated using the same method. Having made these rotations, candidate interatomic distances and candidate interaction energies can be determined and used to obtain p^{ext} for this candidate solution. One of the candidates is probabilisticly selected at 2104, and the backbone and any side chains are rotated according to this candidate into the proposed structure. The Jacobian of this transformation is determined at 2106 by equation 35. Lastly the old acceptance weight and Jacobian are determined at 2107. From the weights and Jacobians the move acceptance probability is found for use at 1805.

Fig. 19B details the determination 2107 of W^{old} and J^{old} for the reverse move from the proposed to the current side chain structure. Temporarily the proposed structure is used as the basis for energy determination at 2008, and the current structure is restored at 2016, when this process is finished. At 2109, a set of K'-1 candidate torsional angles is generated for the selected backbone bond according to p^{int} using the rejection method and the current torsional angle is added to this set. If as preferred, K' is 1, this step results in a set with only the current angle. At 2111, similarly to 2102, the permitted torsional rotations about adjacent backbone bonds are determined from the equations expressing the concerted rotation constraints. Special care is taken to ensure that the original conformation is found by the root finding proc dure. In particular, the s arch interval is c ntered on the known original ϕ_1 and is made as small as necessary to isolate the root, which may be as small as 0.004° or smaller. The current structure must be among these solutions, since it exists. Select it at 2112. W^{old} is computed from the candidate angle solution, making the candidate rotations and determining candidate interactions. Also the Jacobian, J^{old}, of the transformation is computed from the proposed to the current structure.

5.8. CONSENSUS STRUCTURE TEST

Having selected a candidate pharmacophore and determined a best possible consensus structure and best possible structures for the remainder of the binder molecules, the consensus test, step 6, tests whether a consensus structure has actually been found. A consensus pharmacophore structure consists of a spatial arrangement of chemically similar groups shared by all the N binders to high accuracy. Since an actual pharmacophore exists, the N specifically binding members of the screened libraries will share the actual structure. However, the remainder of binder molecules will share no other similar structures to such a high accuracy. Therefore, a structure consensus of the N binders is possible only if the candidate pharmacophore is the actual physical pharmacophore responsible for the actual binding. If the candidate selected relates to other parts of the binder molecules, no structure consensus will be found. Further, if the Monte Carlo determination attempts to impose a consensus on parts of the binder molecules that do not share structure, an inconsistent overall structure will be obtained for the remainder of the binder molecules.

Therefore, two preferred consensus tests are applied: one test asks whether a consistent candidate pharmacophore has been obtained, and a second test asks whether consistent structures have been obtained for the remainder of the binder molecules. Both tests have a preferred absolute and a less preferred relative version.

There are two portions for the first t st. First, are all the consensus pharmacophore distanc s obtain d in the N binders within at least a specified distance, preferably approximately 0.25 Å, of each other? Second, is the consensus energy, $H_{consensus}$, relatively small compared to the total molecular energy (*e.g.*, less than at most approximately 5-10% of the total molecular energy) as determined by the Monte Carlo method?

There are also two portions of the second test. First, can the intramolecular distances predicted by the Monte Carlo method be confirmed by additional distance measurements? Second, since the Monte Carlo method utilizes distance constraints previously measured, one or more of these measurement constraints can be ignored and the predicted distance checked against that measured distance. Tolerances for these tests are distance agreements of at least specified distances, *e.g.*, approximately 0.5 Å, in each binder.

The two preferred tests have been described in the absolute version as requiring checks against absolute tolerances. Alternatively, the values of the pharmacophore distance differences among the binders, $H_{consensus}$, and the differences of the predicted and measured distances can be accumulated for all the possible candidate pharmacophores, the candidate selected being that one minimizing these departures. Therefore, the selected candidate will have the minimum values for the differences of the pharmacophore distances in the binders, the minimum value for $H_{consensus}$, and the minimum values of the differences of predicated from measured distances.

This invention is adaptable to other tests that evaluate the consistency of the consensus structure obtained for the candidate pharmacophore and the accuracy of the structure obtained for the remainder of the binder molecules.

5.9. LEAD COMPOUND DETERMINATION

Having started at step 1 with a target of interest, upon completion of step 6 of Fig. 1 a high resolution pharmacophore structure has been determined as well as supporting structures of the N binder peptides. This high resolution structure is used in step 7 to determine lead compounds for use as a drug that will bind to the original target of interest.

Thus, one or more lead compounds are determined, that share a pharmacophore specification with the determined consensus pharmacophore structure. This determination can be preferably done by one of several methods: by a search of a database of potential drug compounds or of chemical structures (*e.g.*, the Standard Drugs File (Derwent Publications Ltd., London, England), the Bielstein database (Bielstein Information, Frankfurt, Germany or Chicago), and the Chemical Registry (CAS, Columbus, OH)) to identify compounds that contain the pharmacophore specification; by modification of a known lead compound to include the pharmacophore specification; by synthesizing a *de novo* structure containing the pharmacophore specification; or by modification of binders to the target molecule (*e.g.*, isolated in step 2) outside of the pharmacophore structure to render the binder more attractive for use as a drug (*e.g.*, to increase half-life, solubility, ability to achieve desired *in vivo* localization).

Database search queries are based not only on chemical property information but also on precise geometric information. Computer-based approaches rely on database searching to find matching templates; Y.C. Martin, <u>Database searching in drug design</u>, J. Medicinal Chemistry, vol. 35, pp 2145-54 (1992), which is herein incorporated by reference. Existing methods for searching 2-D and 3-D databases of compounds are applicable to this step. Lederle of American Cyanamid (Pearl River, New York) has pioneered molecular shape-searching, 3D searching and trend-vectors of databases. Commercial vendors and other research groups have enhanced searching capabilities [MACSS-3D, Molecular Design Ltd. (San Leandro, CA); CAVEAT, Lauri, G. et al., University of California (Berkeley, CA); CHEM-X, Chemical Design, Inc. (Mahwah, N.J.)].

The pharmacophore structure determined in this invention is adaptable to any of these methods and sources of chemical database searching and to the enumerated non-database methods. Output will be lead compounds suitable for drug design. An important aspect of this invention is that the high resolution pharmacophore structure will lead to highly targeted leads. Lower resolution structures result in a geometric increase in the number of lead compound query matches. Example 1 illustrates this effect.

5.10. APPENDIX: CONCERTED ROTATION

Since the preferred molecules under consideration are conformationally constrained by disulfide bridge(s), a Monte Carlo move that preserves this constraint is required. The "concerted rotation" scheme used for alkanes can be extended to allow rotation of the torsional angles in conformationally constrained peptides. This appendix describes this extension. Dodd et al. (1993) discusses the original, restricted method. (The essential extensions are expressed in equations 27, 28, and 34.) This method is directly applicable to the cyclic residue of proline, and an alternative embodiment of this invention would thermally perturb proline with a move of similar geometric constraints.

Fig. 14 illustrates the geometry under consideration. Illustrated backbone 1600 is a poly-glycine 7-mer. Rigid unit positions are indicated generally by black circles as at 1601 with incoming bonds generally as at 1602. The torsional rotations ϕ_0 to ϕ_6 are about bonds 1602 to 1608, respectively, between sequential, adjacent rigid units. The rigid unit position vectors $\mathbf{r}_0$ to $\mathbf{r}_6$, illustrated as vectors 1610 to 1616, respectively, define the position of these sequential rigid units with respect to a laboratory coordinate system with origin 1609. A C_α rigid unit (B unit) is illustrated in box 1630, and an amide bond (C unit) in box 1631.

To formulate this method, let us consider rotating about seven torsional angles, which will displace the root positions and rotate four rigid units, rotate up to three additional ones, and leave the rest of the peptide fixed. The root position of a rigid unit is the C_α position for a B unit, the C position for a C unit, the C position for a CH_2 unit, and the S position for the S unit in cystine. If unit 5 is a C unit, however, $\underline{r}_5$ is defined to be the backbone amino nitrogen position of that unit. For each unit, let us define θ_i to be the fixed angle between the incoming and outgoing bonds. Thus, $\theta_i = 0$ for a C unit, and $\theta_i = 70.5°$ for all others.

The method leaves the positions $\underline{r}_i$ of units $i \leq 0$ or $i \geq 5$ fixed. The torsion ϕ_0 is changed by an amount $\delta\phi_0$. The values of ϕ_i, $1 \leq i \leq 6$ are then determined so that only the positions r_i of units $1 \leq i \leq 4$ are changed.

The method requires several definitions to present the solution for the new torsional angles. The bond vectors are defined to be the difference in position between unit i and unit i - 1, as seen in the coordinate system of unit i:

$$\underline{l}_i = \underline{r}_i^{(i)} - \underline{r}_{i-1}^{(i)}. \tag{15}$$

Bond vectors $\underline{l}_1$ to $\underline{l}_5$ are illustrated in Fig. 14 at 1620 to 1624, respectively. The length and orientations of the $\underline{l}_i$ are determined by rigid unit structure and the length and angle AMBER parameters for bonds between atom types. The coordinate system of i is such that the incoming bond is along the $\hat{x}$ direction. Thus $\underline{l}_i = l_i \hat{x}$ if atoms r_i and r_{i-1} are directly bonded to each other and has x- and y-components otherwise. Here $\hat{x}$ is a fixed unit vector along the x direction. Now define a rotation matrix that transforms from the coordinate system of unit i+1 to unit i

$$T_i = \begin{pmatrix} \cos\theta_i & \sin\theta_i & 0 \\ \sin\theta_i\cos\phi_i & -\cos\theta_i\cos\phi_i & \sin\phi_i \\ \sin\theta_i\sin\phi_i & -\cos\theta_i\sin_i\phi & -\cos\phi_i \end{pmatrix} \qquad (16)$$

The positions of the units in the frame of unit 1 are, thus, given by:

$$\begin{aligned} \mathbf{r}_1^{(1)} &= \mathbf{l}_1 \\ \mathbf{r}_2^{(1)} &= \mathbf{l}_1 + T_1\mathbf{l}_2 \\ \mathbf{r}_3^{(1)} &= \mathbf{l}_1 + T_1(\mathbf{l}_2 + T_2\mathbf{l}_3) \\ \mathbf{r}_4^{(1)} &= \mathbf{l}_1 + T_1(\mathbf{l}_2 + T_2(\mathbf{l}_3 + T_3\mathbf{l}_4)) \end{aligned} \qquad (17)$$

Further define the matrix that converts from the frame of reference of unit 1 to the laboratory reference frame

$$T_1^{\mathrm{lab}} = [\cos\psi I + nn^T(1-\cos\psi) + M\sin\psi]A. \qquad (18)$$

where

$$M = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix} \qquad (19)$$

and

$$\begin{aligned} \mathbf{n} &= \frac{\hat{\mathbf{x}} \times \mathbf{r}}{|\hat{\mathbf{x}} \times \mathbf{r}|} \\ \cos\psi &= \frac{\mathbf{r} \cdot \hat{\mathbf{x}}}{|\mathbf{r}||\hat{\mathbf{x}}|} \\ \sin\psi &= \frac{|(\mathbf{r} \times \hat{\mathbf{x}})|}{|\mathbf{r}||\hat{\mathbf{x}}|}, \end{aligned}$$

where $\underline{r}$ is the axis of the bond coming into unit 1. The matrix A is a rotation about $\hat{x}$ and is defined so that

$A\underline{l}_1 = \Delta\underline{r}$:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix} \qquad (20)$$

where

$$\begin{aligned} c &= (l_{1y}\Delta r_y + l_{1z}\Delta r_z)/(\Delta r_y^2 + \Delta r_z^2) \\ s &= (-l_{1z}\Delta r_y + l_{1y}\Delta r_z)/(\Delta r_y^2 + \Delta r_z^2). \end{aligned} \qquad (21)$$

Here $\Delta\underline{R} = A[T_1^{1ab}]^{-1}(\underline{r}_1 - \underline{r}_0)$ if unit 0 is a C unit. Otherwise, $\Delta\underline{r} = \underline{l}_1$.

The method proceeds by solving for ϕ_i, $2 \le i \le 6$, analytically in terms of ϕ_1. Then a nonlinear equation is solved numerically to determine which values of ϕ_1, if any, are possible for the chosen value of ϕ_0.

The derivation proceeds in the coordinate system of unit 1, after it has been rotated by the chosen ϕ_0. Define

$$\underline{t} = \underline{r}_5^{(1)} - \underline{l}_1 = [T_1^{1ab}]^{-1}(\underline{r}_5 - \underline{r}_0) - \underline{l}_1. \qquad (22)$$

If $\theta_3 \neq 0$ and $\theta_5 \neq 0$, one can see from Fig. 14 that the distance between unit 3 and unit 5 is known and equal to

$$q_1^2 = \begin{matrix} (l_{4x}\cos\theta_4 - l_{4y}\sin\theta_4 + l_{5x})^2 + \\ (l_{4x}\sin\theta_4 + l_{4y}\cos\theta_4 + l_{5y})^2 \end{matrix} \qquad (23)$$

But this distance can also be written as

$$\begin{aligned} q_1^2 &= |\underline{x} - T_2 \underline{l}_3|^2 \\ \underline{x} &= T_1^{-1}\underline{t} - \underline{l}_2. \end{aligned} \qquad (24)$$

Equating these two results, two values of ϕ_2 are possible

$$\begin{aligned} \phi_2^{I} &= \arcsin(c_1) - \arctan(x_y/x_z) - H\,(x_z) \\ \phi_2^{II} &= \pi - \arcsin(c_1) - \arctan(x_y/x_z) - H\,(x_z), \end{aligned} \qquad (25)$$

with

$$H(x) = \begin{cases} 0, & x>0 \\ \pi, & x<0 \end{cases} \qquad (26)$$

The constant c_1 is given by

$$c_1 = \begin{cases} \dfrac{q_1^2 - x^2 - l_3^2 + 2x_x(\cos\theta_2 l_{3x} + \sin\theta_2 l_{3y})}{-2(\sin\theta_2 l_{3x} - \cos\theta_2 l_{3y})\,(x_y^2 + x_z^2)^{1/2}}, & \theta_3 \neq 0, \theta_5 \neq 0 \\[2ex] \dfrac{l_{3x} + l_{4x} + l_{5x}\cos\theta_4 - x_x\cos\theta_2}{\sin\theta_2\,(x_y^2 + x_z^2)^{1/2}}, & \theta_3 = 0, \theta_5 \neq 0 \\[2ex] \dfrac{(\underline{r}_5 - \underline{r}_2)\cdot(\underline{r}_6 - \underline{r}_5)/l_6 - l_5 - l_{4x}\cos\theta_4 - x_x(\cos\theta_2 l_{3x} + \sin\theta_2 l_{3y})}{(\sin\theta_2 l_{3x} - \cos\theta_2 l_{3y})\,(x_y^2 + x_z^2)^{1/2}}, & \theta_3 \neq 0, \theta_5 = \\[2ex] \dfrac{l_{3x}\cos\theta_4 - x_x(\cos\theta_2 l_{3x} + \sin\theta_2 l_{3y})}{(\sin\theta_2 l_{3x} - \cos\theta_2 l_{3y})\,(x_y^2 + x_z^2)^{1/2}}, & \theta_3 = 0, \theta_5 = 0 \end{cases} \qquad (27)$$

where $\underline{x}$ is given by Eqn. 24 if $\theta_5 \neq 0$, and $\underline{x} = T_1^{-1}[T_1^{lab}]^{-1}(\underline{r}_6 - \underline{r}_5)/l_6$ if $\theta_5 = 0$. Clearly for there to be a solution $|c_1| \leq 1$. The last three equations for c_1 were determined by conditions similar to equating Eqns. 23 and 24. For $\theta_3 = 0$, $\theta_5 \neq 0$, the x component of $\underline{r}_5^{(3)} - \underline{r}_3^{(3)}$ is known to be equal to $(l_{4x} + l_5\cos\theta_4)$. For $\theta_3 \neq 0$, $\theta_5 = 0$, the x component of $\underline{r}_5^{(5)} - \underline{r}_3^{(5)}$ is known to be equal to $l_{5x} + l_{4x}\cos\theta_4$. For $\theta_3 = 0$, $\theta_5 = 0$, the angle between $\underline{r}_3 - \underline{r}_2$ and $\underline{r}_6 - \underline{r}_5$ is known to be equal to θ_4.

To determine ϕ_3 two expressions for $|\mathbf{r}_5 - \mathbf{r}_4|^2$ are again equated to determine that:

$$c_2 = \frac{l_5^2 - y^2 - l_4^2 + 2y_x(\cos\theta_3 l_{4x} + \sin\theta_3 l_{4y})}{2(\sin\theta_3 l_{4x} - \cos\theta_3 l_{4y})(y_y^2 + y_z^2)^{1/2}} \tag{28}$$

$$\begin{aligned} \phi_3^I &= \arcsin(c_2) - \arctan(y_y/y_z) - H(y_z) \\ \phi_3^{II} &= \pi - \arcsin(c_2) - \arctan(y_y/y_z) - H(y_z), \end{aligned} \tag{29}$$

where $\underline{y} = T_2^{-1}(T_1^{-1}\underline{r} - \underline{l}_2) - \underline{l}_3$. . Again, $|c_2| \leq 1$ for there to be a solution.

If $\theta_5 \neq 0$, the value of ϕ_4 can now be determined from:

$$\underline{r}_5^{(1)} = \underline{r}_4^{(1)} + T_1T_2T_3T_4\underline{l}_5. \tag{30}$$

Defining

$$\underline{q}_3 = T_3^{-1}T_2^{-1}T_1^{-1}[T_1^{1ab}]^{-1}(\underline{r}_5 - \underline{r}_4). \tag{31}$$

the equations that define ϕ_4 are given by

$$\begin{aligned} q_{3y} &= \cos\phi_4(\sin\theta_4 l_{5x} - \cos\theta_4 l_{5y}) \\ q_{3z} &= \sin\phi_4(\sin\theta_4 l_{5x} - \cos\theta_4 l_{5y}) \end{aligned} \tag{32}$$

This is a successful rotation if the position of $\underline{r}_6$ is successfully predicted. That is, the equation

$$\underline{r}_6^{(1)} - \underline{r}_5^{(1)} = T_1T_2T_3T_4T_5\underline{l}_6 = [T_1^{1ab}]^{-1}(\underline{r}_6 - \underline{r}_5). \tag{33}$$

must be satisfied. Consider the x-component, which implies

$$F_5(\phi_1) = \begin{cases} (\underline{r}_6^{(1)} - \underline{r}_5^{(1)})^T T_1 T_2 T_3 T_4 \hat{x} - (l_{6x}\cos\theta_5 + l_{6y}\sin\theta_5) = 0, \ \theta_5 \neq 0 \\ (\underline{r}_4 - \underline{r}_3)\cdot(\underline{r}_6 - \underline{r}_5) - l_4 l_6 \cos\theta_4 = 0, \ \theta_3 \neq 0, \theta_5 = 0 \\ |\underline{r}_6 - \underline{r}_4| - [(l_{6x} + l_{5x})^2 + l_{5y}^2]^{1/2} = 0, \ \theta_3 = 0, \theta_5 = 0 \end{cases} \quad (34)$$

must be satisfied if the rotation is successful. The equations for the case $\theta_5 = 0$ clearly express the geometric conditions required for a successful rotation.

Eqn. 34 is the nonlinear equation for ϕ_1 because ϕ_2, ϕ_3, and ϕ_4 are determined by Eqns. (25), (29), and (32) in terms of ϕ_1. This equation has between zero and four values for each value of ϕ_1, however, due to the multiple root character of Eqns. (25) and (29). The equation is solved by searching the region $-\pi < \phi < \pi$ for zero crossings. The search is in increments of $\sim 0.04°$. These roots are then refined by a bisection method.

The transformation from ϕ_i, $0 \leq i \leq 6$ to the new solution which is constrained to change only r_i, $1 \leq i \leq 4$ actually implies a change in volume element in torsional angle space. This change in volume element is the reason for the appearance of the Jacobian in the acceptance probability. The Jacobian of this transformation is calculated in Dodd et al. (1993)at pp. 991-93. It is slightly different here since root position $\underline{r}_5$ is not necessarily the head position. The Jacobian is given by.

$$J = \frac{1}{|\det B|} \quad (35)$$

where the 5 x 5 matrix B is given by $B_{ij} = [\underline{u}_j \times (\underline{r}_5 - \underline{h}_j)]_i$ for i

≤ 3 and $B_{ij} = [\underline{u}_j \times (\underline{r}_6 - \underline{r}_5)/|\underline{r}_6 - \underline{r}_5|]_{i-3}$ for $i = 4,5$. Here $\underline{h}_i = \underline{r}_i$, except that $\underline{h}_5$ is the head position even if $\theta_5 = 0$, and $\underline{u}_i$ is the incoming bond vector for unit i.

Repeated application of the concerted rotation may lead to a slightly imperfect structure, due to numerical precision errors. In an alternative embodiment, peptide geometry would be restored to an ideal state by application of the Random Tweek algorithm after several thousand moves (Shenkin et al., 1987, Biopolymers 26:2053-85).

The invention is further described in the following examples which are in no way intended to limit the scope of the invention.

6. EXAMPLES

6.1. RELATION BETWEEN EFFECTIVENESS OF POTENTIAL DRUG IDENTIFICATIONS AND PHARMACOPHORE GEOMETRIC TOLERANCE

Searches of a drug library well known to medicinal chemists, the Standard Drugs File (Derwent Publications Ltd., London, England), illustrate the geometric increase in the number of compounds found (and thus decrease in expected effectiveness of identification of potential drugs) as pharmacophore geometric tolerance is increased. Table 4 tabulates the results.

Table 4

5HT3 (5 Hydroxytryptophan)	
Tolerance (Å)	Number of drug compounds
2.0	64
1.0	35
0.5	27
0.25	12
0.10	1

Dopamine	
Tolerance (Å)	Number of drug compounds
2.0	188
1.0	185
0.5	60
0.25	48
0.10	5

The pharmacophores are two well known neurotransmitters, 5-hydroxytryptophan and dopamine. As the tolerance of one distance in the pharmacophore structure is decreased from 2.0 to 0.1 Å, the number of compounds retrieved from the database is listed. The advantage of achieving pharmacophore resolution better than approximately 0.25 Å is clear.

If the tolerance of three distances were involved, the expected number of compound retrieved would be the cube of these numbers. For the dopaminergic pharmacophore, the number of lead compounds would decrease from over 6.5×10^6 to about 125 as three tolerances were decreased from 2.0 Å to 0.1 Å.

This example illustrates the geometric increase in the number of leads identified as pharmacophore geometry is less well defined. It thus a very preferred aspect of this invention that the computational method results in determining pharmacophore structure accurate to at least approximately 0.25 to 0.30 Å. Thus an exponentially large improvement in lead compound selection for drug design can be expected to result from this invention.

6.2. EXPRESSION AND PURIFICATION OF TARGET PROTEINS

Target molecules that are proteins, for example ras, raf, vEGF and KDR, are expressed in the *Pichia pastoris* expression system (Invitrogen, San Diego, CA) and as glutathione-S-transferase (GST)-fusion proteins in *E. coli* (Guan and Dixon, 1991, Anal. Bioch m. 192:262-267).

The cDNAs of these target proteins are cloned in the *Pichia* expression vectors pHIL-S1 and pPIC9 (Invitrogen). Polymerase chain reaction (PCR) is used to introduce six Histidines at the carboxy-terminus of these proteins, so that this His-tag can be used to affinity-purify these proteins. The recombinant plasmids are used to transform *Pichia* cells by the spheroplasting method or by electroporation. Expression of these proteins is inducible in *Pichia* in the presence of methanol. The cDNAs cloned in the pHIL-S1 plasmid are expressed as a fusion with the PHO1 signal peptide and hence are secreted extracellularly. Similarly cDNAs cloned in the pPIC9 plasmid are expressed as a fusion with the α-factor signal peptide and hence are secreted extracellularly. Thus, the purification of these proteins is simpler as it merely involves affinity purification from the growth media. Purification is further facilitated by the fact that *Pichia* secretes very low levels of homologous proteins and hence the heterologous protein comprises the vast majority of the protein in the medium. The expressed proteins are affinity purified onto an affinity matrix containing nickel. The bound proteins are then eluted with either EDTA or imidazole and are further concentrated by the use of centrifugal concentrators.

As an alternative to the *Pichia* expression system, the target proteins are expressed as glutathione-S-transferase (GST) fusion proteins in *E. coli*. The target protein cDNAs are cloned into the pGEX-KG vector (Guan and Dixon, 1991, Anal. Biochem. 192:262-267) in which the protein of interest is expressed as a C-terminus fusion with the GST protein. The pGEX-KG plasmid has an engineered thrombin cleavage site at the fusion junction that is used to cleave the target protein from the GST tag. Expression is inducible in the presence of IPTG, since the GST gene is under the influence of the tac promoter. Induced cells are broken up by sonication and the GST-fusion protein is affinity purified onto a glutathione-linked affinity matrix. The bound protein is then cleaved by the addition of thrombin to the affinity matrix and recovered by washing, while the GST tag remains bound to the matrix. Milligram quantities of recombinant protein per liter of *E. coli* culture are expected to be obtainable in this manner.

6.3. SYNTHESIS AND SCREENING OF POLYSOME-BASED LIBRARIES ENCODING RANDOM CONSTRAINED PEPTIDES OF VARIOUS LENGTHS

6.3.1. PREPARATION OF DNA TEMPLATES

DNA libraries with a high degree of complexity are made as two components: an expression unit, and a semi-random (or degenerate) unit. The expression unit has been synthesized chemically as an oligonucleotide (termed T7RBSATG), and contains the promoter region for bacteriophage T7 RNA polymerase, a ribosome binding site, and the initiating ATG codon. The random region, also synthesized as an oligonucleotide (termed MMN6) contains a region complementary to the expression unit, the antisense version of the codons specifying Cys-X_6-Cys, and a restriction site (BstXI). The library is constructed by annealing 100 pmol of oligonucleotide T7RBSATG [having the sequence 5'ACTTCGAAATTAATACGACTCACTATAGGGAGACCACAACGGTTTCCCTCCAGAAAT AATTTTGTTTAACTTTAACTTTAAGAAGGAGATATACATATGCAT3' (SEQ ID NO:2)]; and oligonucleotide MNN6 [having the sequence 5'CCCAGACCCGCCCCCAGCATTGTGGGTTCCAACGCCCTCTAGACA[MNN]$_6$ACAATG TATATCTCCTTCTT3' (SEQ ID NO:3); M = A or C , N = G, A, T, or C], and extending the DNA in a reaction mixture containing 10-100 units of Sequenase (United States Biochemical Corp., Cleveland, OH), all four dNTPS (at 1 mM), and 10 mM dithiothreitol for 30 min at 37°C. The extended material is then digested with BstXI, ethanol precipitated and resuspended in water. This fragment of DNA is then ligated via the BstXI end to a 250 base pair (bp), PCR-amplified Glycine-Serine coding fragment derived from gene III of M13 bacteriophage DNA. The gene III fragment has been amplified by use of two primers, respectively termed FGSPCR [having the sequence 5'TCGTCTGACCTGCCTCAACCTCCCCACAATGCTGGCGGCGGCTCTGGT3'

(SEQ ID NO: 4)], and RGSPCR [having the sequence 5'ATCAAGTTTGCCTTTACCAGCATTGTGGAGCGCGTTTTCATC3' (SEQ ID NO:5)], and Taq DNA polymerase (Gibco-BRL). The amplified DNA (250 bp) was cut with BstXI to yield a 200 bp fragment that has been gel purified. The 200 bp fragment is then ligated to the random peptide coding DNA fragment. This DNA specifies the synthesis of a peptide of the sequence Met-His-Cys-$(X)_6$-Cys- (SEQ ID NO:6) fused to the Gly-Ser rich region of the M13 gene III protein. The Gly-Ser rich domain is thought to behave as a flexible linker and assist in presentation of the random peptide to the target molecules.

To make constrained random peptides of different lengths, oligonucleotides are made that are similar to MNN6, except that the degenerate region is 5, 7, 8, and 9 codons long. In addition, oligonucleotides are made that code for various shapes of constrained random peptides by specifying sequences comprising three cysteine residues interspersed between 6-10 randomly specified amino acids.

6.3.2. *IN VITRO* SYNTHESIS AND ISOLATION OF POLYSOMES

An *E. coli* S30 extract is prepared from the B strain SL119 (Promega). Coupled transcription-translation reactions are performed by mixing the S30 extract with the S30 premix (containing all 20 amino acids), the linear DNA template coding for peptides of random sequences (prepared as described in Section 6.3.1 above), and rifampicin at 20 µg/ml. The reaction is initiated by the addition of 100 units of T7 RNA polymerase and continues at 37°C for 30 min. The reaction is terminated by placing the reactions on ice and diluting them 4-fold with polysome buffer (20 mM Hepes-NaOH, pH 7.5, 10 mM $MgCl_2$, 1.5 µg/ml chloramphenicol, 100 µg/ml acetylated bovine serum albumin, 1 mM dithiothreitol, 20 units/ml RNasin, and 0.1% Triton X-100). Polysomes are isolated from a 50 µl reaction programmed with 0.5-1 µg of linear DNA template specifying the synthesis of random constrained peptides. To isolate polysomes, the diluted S30 reaction mixtures are centrifuged at 288,000 X g for 30-40 min at 4°C. The pellets are suspended in polysome buffer and centrifuged a second time at 10,000 X g for 5 min to remove insoluble material.

6.3.3. AFFINITY SELECTION/SCREENING OF POLYSOMES

The isolated polysomes are incubated in microtiter wells coated with the target proteins. Microtiter wells are uniformly coated with 1-5 μg of 6-His tagged, or glutathione S-transferase fused, target proteins (see Section 6.2 hereinabove). Target proteins that are used include the oncoproteins ras and raf, KDR (the vascular endothelial growth factor [vEGF] receptor protein) and vEGF. The microtiter wells are coated with 1-5 μg of these target proteins by incubation in PBS (phosphate-buffered saline; 10 mM sodium phosphate, pH 7.4, 140 mM NaCl, 2.7 mM KCl), for 1-5 hours at 37°C. The wells are then washed with PBS, and the unbound surfaces of the wells blocked by incubation with PBS containing 1% nonfat milk for 1 hr at 37°C. Following a wash with polysome buffer, each well is incubated with polysomes isolated from a single 50 μl reaction for 2-24 hr at 4°C. Each well is washed five times with polysome buffer and the associated mRNA is eluted with polysome buffer containing 20 mM EDTA.

After affinity selection of the polysomes, the associated mRNAs are isolated, and treated with 5-10 units of DNase I (RNase-free; Ambion) for 15 min at 37°C after addition of $MgCl_2$ to 40 mM. The mRNA is phenol-extracted and ethanol-precipitated and dissolved in 20 μl of RNase-free water. A portion of the mRNA is used for cDNA preparation and subsequent amplification using 15 pmol each of primers RGSPCR [5'ATCAAGTTTGCCTTTACCAGCATTGTGGAGCGCGTTTTCATC3' (SEQ ID NO:5)], and SELEXF1 [5'ACTTCGAAATTAATACGACTCACTATAGGGAGACCACAACGGTTTCC3' (SEQ ID NO:9)] and rTth Reverse Transcriptase RNA PCR kit (Perkin Elmer Cetus). Specifically, the mRNA is reversetranscribed into cDNA in a 20 μl reaction containing 1 pg mRNA, 15 pmol of RGSPCR primer, 200 μM each of dGTP, dATP, dTTP, and dCTP, 1 mM $MnCl_2$, 10 mM Tris-HCl, pH 8.3, 90 mM KCl, and 5 units of rTth DNA polymerase at 70°C for 15 min. In the next step, the cDNA is amplified by the addition of 2.5 mM $MgCl_2$, 8% glycerol, 80 mM Tris-HCl, pH 8.3, 125 mM KCl, 0.95 mM EGTA, 0.6% Tween 20, and 15 pmol of the SELEXF1 primer. The reaction conditions that are employed are 2 min at 95°C for one cycle, 1 min at 95°C and 1 min at 60°C for 35 cycles, and 7 min at 60°C for one cycle. The amplified product is then gel-purified and quantitated by spectrophotometry at 260 nm. A portion of the amplified DNA is digested with NsiI and XbaI and the resulting 30 base pair fragment is directionally cloned into a monovalent phage display vector. The DNAs inserted in the monovalent phage display vector are then sequenced to determine the identity of the peptides that were selectively retained by one cycle of affinity binding to the target protein. A second portion (0.5-1 μg) of the amplified DNA is subjected to another cycle of affinity selection, mRNA isolation, cDNA amplification, and cloning.

6.4. <u>PHAGEMID SCREENING</u>

Three different protocols for screening of a phagemid library are presented in the subsections hereinbelow. These protocols, particularly the immobilization and binding steps, are readily adaptable to use for screening of different libraries, *e.g.*, polysome libraries. Preferably, different methods are used in different rounds of screening.

6.4.1. <u>PLATE PROTOCOL</u>

In this example, a protocol is presented for screening a phagemid library, in which in the first round of screening, a biotinylated target protein is immobilized (by the specific binding b tween biotin and streptavidin) on a streptavidin coated plate. The immobilized target protein is then contacted with library members to select binders.

Reagents Used:
Purified target protein, microfuge tubes, Falcon 2059, Binding Buffer, Wash Buffer, Elute Buffer, phage display Library of $>10^{11}$ pfu/Screened Target, fresh overnight cultures of appropriate host cells, LB Agar plates with antibiotics as needed, biotinylating agent NHS-LC-Biotin (Pierce Cat. #21335), streptavidin, 50 mM $NaHCO_3$ pH 8.5, 1 M Tris pH 9.1, M280 Sheep anti-mouse IgG coated Dynabeads (Dynal), phosphate buffered saline (PBS), Falcon 1008 petri dishes.

Wash Buffer = 1X PBS (Sigma Tablets), 1 mM $MgCl_2$, 1 mM $CaCl_2$, 0.05% Tween 20; (For one liter: 5 PBS tablets, 1 ml 1 M $MgCl_2$, 1 ml 1 M $CaCl_2$, 0.5ml Tween 20, nanopure H_2O to 1 liter).

Binding Buffer = Wash Buffer with 5 mg/ml bovine serum albumin (BSA).

Elute Buffer = 0.1 N HCl adjusted to pH 2.2 with glycine: 1 mg/ml BSA.

Procedure:
Protein Biotinylation:
1. Wash 50-100 μg of target protein in 50 mM $NaHCO_3$ pH 8.5 in a Centricon (Amicon) of the appropriate molecular weight cut-off.
2. Bring the total volume to 100 μl with 50 mM $NaHCO_3$ pH 8.5.
3. Dissolve 1 mg of NHS-LC-Biotin in 1 ml H_2O. Do not store this solution.
4. Immediately add 37 μl of the NHS-LC-Biotin solution to the target protein and incubate for 1 hr at room temperature (RT).

5. Remove the unreacted biotin by washing 2X PBS in a Centricon (Amicon) of the appropriate molecular weight cutoff. Store the biotinylated protein at 4°C.

Coating a 1008 Plate with Streptavidin:

6. The night before the binding experiment precoat a 1008 plate with streptavidin.

7. Add 10 μg of streptavidin (1 mg/ml H_2O) per 1 ml of 50 mM $NaHCO_3$ pH 8.5.

8. Add 1 ml of this solution to each plate and place in a humidified chamber overnight at 4°C.

Prebinding; Blocking Non-Specific Sites:

9. To a streptavidin coated plate add 400 μl of Binding Buffer (BSA blocking) for one hour at room temperature.

10. Rinse wells six times with Wash Buffer by slapping dry on a clean piece of labmat.

Binding; Specific Target/Phage Complexes Round 1:

11. Add 10 μg of biotinylated target protein in 400 μl of Binding Buffer to the well and incubate for 2 hr at 4°C.

12. Add 4 μl of 10 mM biotin and swirl for 1 hr at 4°C.

13. Wash as in step 10.

14. Add concentrated phage library ($>10^{11}$ pfu) in 400 μl of Binding Buffer and swirl overnight at 4°C.

Washing and Elution:

15. Slap out binding mixture and wash as in step 10.

16. To elute bound phage add 400 μl of Elution Buffer and rock at RT for 15 min.

17. Transfer the elution solution to a sterile 1.5 ml tube which contains 75 μl of 1 M Tris pH 9.1. Vortex briefly.

Amplification of Round 1 Eluted Phage:

18. Plate all of the eluted round 1 phage by adding 157 μl cf phage to 200 μl of cells incubated overnight (previously ch cked free of contamination) in three aliquots. Incubate

25 min in a 37°C water bath and then spread onto LB agar/antibiotics plate containing 2% glucose.

19. Scrape plates with 5 ml of 2XYT (growth broth)/ Antibiotics/Glucose and leave swirling for 30 min at RT.

20. Add the appropriate amount of 2XYT/Antibiotics/Glucose to bring the O.D. 600 down to 0.4 and then grow at 37°C at 250 rpm until the O.D. 600 reaches 0.8.

21. Remove 5 ml and add to it 1.25 x 10^{10} M13 helper phage.

22. Shake 30 min at 150 rpm and then 30 min at 250 rpm at 37°C.

23. Centrifuge 10 min at 3000 X g at RT.

24. Resuspend cells in 5 ml 2XYT with no glucose. (This step removes glucose).

25. Centrifuge as in step 23 and resuspend in 5 ml 2XYT with kanamycin and the appropriate antibiotics (no glucose). Spin 18 hr at 37°C and 250 rpm.

26. Pellet cells at 10,000 X g and sterile filter the phage containing supernatant which is now ready for round 2 screening.

27. Titer the round 1 eluted phage stocks.

Binding; Specific Target/Phage Complexes Rounds 2-5:

6. Combine ~1 μg of biotinylated target protein with the eluted and titered round 1 phage (10^9 pfu) in 200 μl of Binding Buffer and rock 4 hr at 4°C.

7. The night before the round 2 screening is started, prewash 200 μl/target protein to be screened of sheep anti-mouse IgG magnetic beads (M280 IgG Dynabeads) with 2X 1 ml of Wash Buffer using the Dynal Magnet. Let the beads collect at least 1 min before removing the buffer. Let the beads stand 15 sec to allow residual Binding Buffer to collect and remove with a P200 Pipetman.

8. Resuspend the washed beads in 200 μl of Binding Buffer and add 100 μl of mouse anti-biotin IgG (Jackson IRL). Rock overnight at 4°C.

10. Wash the unbound anti-biotin IgG from the Dynabeads by placing th m on the Dyna magnet for at least 1 min and remove

all liquid as in Step 7. Remove the tube from the magnet and resuspend the beads in 1 ml of Wash Buffer, rock at 4°C for 30 min, and return to the magnet. Again let the beads pellet for 1 min; repeat this process 3 more times and resuspend the beads in 400 µl of Binding Buffer.

10a. The coated beads are now ready for use (100 µl/round/target protein). The remainder can be stored for use for up to 2 weeks.

11. Add the 100 µl of anti-biotin coated Dynabeads (Step 10) to the protein/phage fraction (Step 9) bringing the total binding volume to 300 µl and rock for 2 hr at 4°C. Ensure that the beads mix thoroughly with the phage/protein solution.

Washing and Elution:

12. Place the binding reaction into the Dynal magnet and let sit for 1 min.

13. Remove the solution using a P1000 Pipetman and discard. Let the beads stand 15 sec to allow residual binding buffer to collect and remove with a P200 Pipetman. Note serial dilution depends upon all residual liquid being removed (i.e., 5 µl into 500 is 100X washing; 50 µl into 500 is only 10X).

14. Remove the tube from the magnet and resuspend the beads in 750 µl of Wash Buffer and return to the magnet. Again let the beads pellet by waiting 1 min.

15. Remove the Wash solution as in Step 7 and repeat this process several more times.

16. After the removal of the final wash, resuspend the beads and transfer them to a fresh, labeled tube and wash once more.

17. To elute bound phage, add 400 µl of Elution Buffer, titrate and rock for 14 min at RT.

18. Place the tube on the magnet for one minute and transfer the eluate to a sterile 1.5 ml tube which contains 75 µl of 1 M Tris pH 9.1. Vortex briefly.

Amplification of Round 2-5 Eluted Phage:
15a. Plate 10 μl and 100 μl of round 2,3,4 eluates using 200 μl of contamination free (previously tested) *E. coli* XL1Blue cells onto each plate containing tetracycline/ampicillin/glucose and tetracycline/ampicillin and amplify as in Steps 17-25.

6.4.2. BIOTIN-ANTIBIOTIN IgG BEAD PROTOCOL

In this example, a protocol is presented for screening a phagemid library, in which a biotinylated target protein is immobilized (by the specific binding between anti-biotin antibodies and biotin) on a magnetic bead containing anti-biotin antibodies on the bead surface. The immobilized target protein is then contacted with library members to select binders.

Reagents Used:
M280 Sheep anti-Mouse IgG coated Dynabeads (Dynal)

Binding; Specific Target/Phage Complexes Round 1:
6. Combine 10 μg of biotinylated target protein with the phage library (>10^{10} pfu) in 400 μl of Binding Buffer and rock overnight at 4°C.
7. That same night prewash 50 μl sheep anti-mouse IgG magnetic beads (M280 IgG Dynabeads) with 500 μl of Binding Buffer twice using the Dynal Magnet. Let the beads collect at least 1 min before removing the buffer. Let the beads stand 15 sec to allow residual binding buffer to collect and remove with a P200 Pipetman.
8. Resuspend the washed beads in 100 μl of Binding Buffer and add 33 μl of mouse anti-biotin IgG (40 μg, Jackson IRL). Rock overnight at 4°C.
9. Remove unbound protein from the phage/protein reaction in Step 6 with a Microcon 100. Spin at 800 X g until exclusion volume is met and wash twice with Wash Buffer (again at 800 X g). Collect phage/protein with a Pipetman and add an additional 50 μl of Wash Buffer to the Microcon,

gently titrate and combine with first fraction to ensure maximal recovery.

10. Wash the unbound anti-biotin IgG from the Dynabeads by placing them on the Dyna magnet for at least 1 min and remove all liquid as in Step 7. Remove the tube from the magnet and resuspend the beads in 750 µl of Wash Buffer, rock at 4°C for 30 min, and return to the magnet. Again, let the beads pellet for 1 min; repeat this process 3 more times and resuspend the beads in 100 µl of Binding Buffer.

11. Add the anti-biotin coated Dynabeads (Step 10) to the protein/phage fraction (Step 9), bring the total binding volume to 500 µl with Binding Buffer, and rock for 2 hr at RT. Ensure that the beads mix thoroughly with the phage/protein solution.

Washing and Elution:

12. Place the binding reaction into the Dynal magnet and let sit for 1 min.

13. Remove the solution using a P1000 Pipetman and discard. Let the beads stand 15 sec to allow residual binding buffer to collect and remove with a P200 Pipetman. Note that serial dilution depends upon all residual liquid being removed (i.e., 5 µl into 500 is 100X washing; 50 µl into 500 is only 10X).

14. Remove the tube from the magnet and resuspend the beads in 750 µl of Wash Buffer and return to the magnet. Again let the beads pellet by waiting 1 min.

15. Remove the wash solution as in Step 7 and repeat this process 3 more times.

16. After the removal of the fourth wash, resuspend the beads and transfer them to a fresh, labeled tube and wash once more.

17. To elute bound phage, add 400 µl of Elution Buffer, titrate and rock for 14 min at RT.

18. Place the tube on the magnet for one minute and transfer the eluate to a sterile 1.5 ml tube which contains 75 µl of 1 M Tris pH 9.1. Vortex briefly.

Amplification of Round 1 Eluted Phage:

17. Plate all of the eluted round 1 phage by adding 157 μl of phage to 200 ml of cells incubated overnight (previously checked to be free of contamination) in three aliquots. Incubate 25 min in a 37°C water bath and then spread onto LB agar/antibiotics plate containing 2% glucose. Place plates upright in 37°C incubator until dry and then invert and incubate overnight.

18. Scrape plates with 5 ml of 2XYT/Antibiotics/Glucose and leave swirling for 30 min at RT.

19. Add the appropriate amount of 2XYT/Antibiotics/Glucose to bring the O.D. 600 down to 0.4 and then grow at 37°C at 250 rpm until the O.D. 600 reaches 0.8.

20. Remove 5 ml and add to it 1.25 x 10^{10} M13 helper phage.

21. Shake 30 min at 150 rpm and then 30 min at 250 rpm at 37°C.

22. Centrifuge 10 min at 3000 X g at RT.

23. Resuspend cells in 5 ml 2XYT with no glucose. (This step removes glucose)

24. Centrifuge as in step 23 and resuspend in 5 ml 2XYT with kanamycin and the appropriate antibiotics (no glucose). Spin 18 hr at 37°C and 250 rpm.

25. Pellet cells at 10,000 xg and sterile filter the phage-containing supernatant which is now ready for round 2 screening.

Binding; Specific Target/Phage Complexes Round 2, 3, & 4:

6a. Bind 1 μg of target protein with 100 μl of amplified phage from the previous round as before, overnight at 4°C.

7a. Prepare the IgG anti biotin/anti IgG beads as in Steps 7-10 using, however, only 20 μl of sheep anti-mouse IgG and 13 μl of anti-biotin IgG.

8a. All other binding procedures are identical with Steps 6-11.

Washing and Elution:

9a. Place the binding reaction into the Dynal magnet and let sit for 1 min.

10a. Remove the solution and discard using a P1000 Pipetman. Let the beads stand 30 sec to allow residual Binding Buffer to collect and remove with a P200 Pipetman.

11a. Remove the tube from the magnet and resuspend the beads in 750 μl of Wash Buffer and return to the magnet. Again let the beads pellet by waiting 1 min.

12a. Remove the wash solution as in Step 11a and repeat this process 3 more times.

13a. After the removal of the fourth wash, resuspend the beads and transfer them to a fresh, labeled tube and wash 4 more times.

14a. Elute and neutralize as in Step 15.

Amplification of Rounds 2, 3, & 4 Eluted Phage:

15a. Plate 10 μl and 100 μl of round 2,3,4 eluates and amplify as in Steps 17-25.

6.4.3. BIOTIN-STREPTAVIDIN, MAGNETIC BEAD PROTOCOLS

In this example, a protocol is presented for screening a phagemid library, in which a biotinylated target protein is immobilized (by the specific binding between biotin and streptavidin) on a streptavidin coated magnetic bead. The immobilized target protein is then contacted with library members to select binders.

Reagents Used:

Purified target protein, M280 streptavidin coated Dynabeads (Dynal)

Binding; Specific Target/Phage Complexes Round 1:

6. Combine 10 μg of biotinylated target protein with the phage library ($>10^{10}$ pfu) in 400 μl of Binding Buffer and rock overnight at 4°C.

7. Remove unbound protein with a Microcon 100. Spin at 800 X g until exclusion volume is met, and wash twice with Wash Buffer (again at 800 X g). Collect phage/protein with a Pipetman and add an addition 50 μl of Wash Buffer to the Microcon, gently titrate and combine with the first fraction to ensure maximal recovery.

8. Prewash 50 μl (per reaction) of streptavidin magnetic beads (M280 streptavidin Dynabeads) twice with 500 μl of Washing Buffer using the Dynal magnet.

9. Add the prewashed Dynabeads to the protein/phage fraction (add Binding Buffer to a total of 500 μl) and rock for 30 min. Ensure that the beads mix thoroughly with the phage/protein solution.

Washing and Elution:

10. Place the binding reaction into the Dynal magnet and let sit for 1 min.

11. Remove the solution using a P1000 Pipetman and discard. Let the beads stand 15 sec to allow residual Binding Buffer to collect and remove with a P200 Pipetman. Note that serial dilution depends upon all residual liquid being removed (*i.e.*, 5 μl into 500 is 100X washing; 50 μl into 500 is only 10X).

12. Remove the tube from the magnet and resuspend the beads in 750 μl of Wash Buffer and return to the magnet. Again let the beads pellet by waiting 1 min.

13. Remove the wash solution as in step 11 and repeat this process 3 more times.

14. After the removal of the fourth wash, resuspend the beads and transfer them to a fresh, labeled tube and wash once more.

15. To elute bound phage add 400 μl of Elution Buffer, titrate and rock for 14 min at RT.

16. Place the tube on the magnet for one minute and transfer the eluate to a sterile 1.5 ml tube which contains 75 μl of 1 M Tris pH 9.1. Vortex briefly.

Amplification of Round 1 Eluted Phag :

17. Plate all of the eluted round 1 phage by adding 157 μl of phage to 200 μl of overnight cells (previously checked to be free of contamination) in three aliquots. Incubate 25 min in a 37°C water bath and then spread onto LB agar/antibiotics plate containing 2% glucose. Place plates upright in 37°C incubator until dry and then invert and incubate overnight.

18. Scrape plates with 5 μl of 2XYT/Antibiotics/Glucose and leave swirling for 30 min at RT.

19. Add the appropriate amount of 2XYT/Antibiotics/Glucose to bring the O.D. 600 down to 0.4 and then grow at 37°C at 250 rpm until the O.D. 600 reaches 0.8.

20. Remove 5 ml and add to it 1.25 x 10^{10} M13 helper phage.

21. Shake 30 min at 150 rpm and then 30 min at 250 rpm at 37°C.

22. Centrifuge 10 min at 3000 X g at RT.

23. Resuspend cells in 5 μl 2XYT with no glucose. (This step removes glucose).

24. Centrifuge as in step 22 and resuspend in 5 ml 2XYT with kanamycin and the appropriate antibiotics (no glucose). Shake 18 hr at 37°C and 250 rpm.

25. Pellet cells at 10,000 X g and sterile filter the phage containing supernatant which is now ready for round 2 screening.

Binding; Specific Target/Phage Complexes Round 2, 3, & 4:

6a. Combine 1 μg of biotinylated target protein with 100 μl of the previous round's phage (>10^9 pfu) in 400 μl of Binding Buffer and rock overnight at 4°C.

7a. Remove unbound protein with a Microcon 100. Spin at 800 X g until exclusion volume is met and wash twice with Wash Buffer (again at 800 X g). Collect phage/protein with a Pipetman and add an addition 50 μl of Wash Buffer to the Microcon, gently titrate and combine with the first fraction to ensure maximal recovery.

8a. Prewash 20 μl (per reaction) of streptavidin magnetic beads (M280 streptavidin Dynabeads) twice with 500 μl of Washing Buffer using the Dynal magnet.

9a. Add the prewashed Dynabeads to the protein/phage fraction and rock for 30 min. Add Binding Buffer to a total of 500 μl. Ensure that the beads mix thoroughly with the phage/protein solution.

Washing and Elution:

10a. Place the binding reaction into the Dynal magnet and let sit for 1 min.

11a. Remove the solution and discard using a P1000 Pipetman. Let the beads stand 30 sec to allow residual Binding Buffer to collect and remove with a P200 Pipetman.

12a. Remove the tube from the magnet and resuspend the beads in 750 μl of Wash Buffer and return to the magnet. Again let the beads pellet by waiting 1 min.

13a. Remove the wash solution as in Step 11a and repeat this process 3 more times.

14a. After the removal of the fourth wash resuspend the beads and transfer them to a fresh, labeled tube and wash 4 more times.

15a. Elute and neutralize as in Step 15.

Amplification of Rounds 2, 3, & 4 Eluted Phage:

16a. Plate 10 μl and 100 μl of round 2,3,4 eluates and amplify as in Steps 17-25.

6.5. AFFINITY MEASUREMENTS OF <u>PEPTIDE-TARGET PROTEIN INTERACTIONS</u>

Once peptides that bind to a target protein have been identified, the affinities of these peptides to their respective targets are measured by measuring the dissociation constants (K_d) of each of these peptides to their respective targets. Oligonucleotides that encode the peptides are constructed so as to encode also an epitope tag fused to the peptide (for example, the myc epitope) that can be detected by a commercially available antibody. These oligonucleotides are incubated with polysome extracts to produce the peptide tagged with the epitope. Binding of the target protein to the peptide is done in solution, and separation of the bound peptide from the unbound peptide is done by immunoaffinity purification using an anti-target protein antibody. This immunoaffinity purification is done by a modified ELISA (enzyme-linked immunosorbent assay) protocol, in which the target protein-peptide mixture is exposed to the anti-target protein antibody immobilized on a solid support such as a nitrocellulose membrane, and the unbound peptide is then washed off. In this protocol, the concentration of the target protein is varied and then the amount of bound peptide is estimated by detecting the epitope tag on the peptide by use of anti-epitope antibody. In this manner, the affinity of each peptide for its target protein can be determined.

6.6. <u>REDOR MEASUREMENTS ON A CX_6C PEPTIDE RESIN</u>

This example demonstrates successful synthesis and cyclization of a CX_6C peptide resin of greater than 95% purity and with a labeled glycine followed by successful REDOR distance measurements on the CX_6C peptide resin using the preferred REDOR methods of this invention. The labeled peptide used was Cys-Asn-Thr-Leu-Lys-(^{15}N-2-^{13}C)Gly-Asp-Cys-Gly-mBHA resin, where a glycine linker attached the peptide of interest to the nBHA resin. (Cys-Asn-Thr-Leu-Lys-Gly-Asp-Cys-Gly = SEQ ID NO:10)

The peptide resin was synthesized by solid phase synthesis on p-MethylBenzhydrilamine (mBHA) resin using a combination of Boc and Fmoc chemistry. MethylBenzhydrilamine resin (Subst. 0.36 meq/g) was purchased from Advanced Chem Tech (Louisville, KY). Fmoc(^{15}N-2-^{13}C)Gly was prepared from HCl, (^{15}N-2-^{13}C)Gly (Isotec Inc., Miamisburg, OH) and Fmoc-OSu. Boc-Gly, (Trt), Fmoc-Asp(OtBu), Fmoc-Lys(Boc), Fmoc-Leu, Fmoc-Thr(OtBu), Fmoc-Asn and Boc-Cys(Acm) were purchased from Bachem (Torrance, CA). Reagent grad solvents were purchased from Fisher Scientific, Diisopropylcarbodiimide (DIC), Trifluoroacetic acid (TFA) and Diisopropylethylamine (DIEA) were purchased from Chem Impex (Wooddale, IL). Nitrogen, HF were purchased from Air Products (San Diego, CA).

The first step 43 was the synthesis of Boc-Cys(ACM)-Asn-Thr(OtBu)-Leu-Lys(Boc)-Gly-Asp(OtBu)-Cys(Trt)-Gly-mBHA resin. 1.11g (0.40 meq) of mBHA resin were placed in a 150 ml reaction vessel (glass filter at the bottom) with Methylene Chloride (CH_2Cl_2) ["DCM"] and stirred 15 min with a gentle bubbling of Nitrogen in order to swell the resin. The solvent was drained and the resin was neutralized with DIEA 5% in DCM (3X2 min). After washes with DCM, the resin was coupled 60 min with Boc-Gly (0.280 g-1.6 meq-4 fold excess-0.1M) and DIC (0.25 ml-1.6 meq-4 fold excess-0.1M) in DCM. Completion of the coupling was checked with the Ninhydrin test. After washes, the resin was stirred 30 min in TFA 55% in DCM in order to remove the Boc protecting group. The resin was then neutralized with DIEA 5% in DCM and coupled with Fmoc-Cys(Trt)(0.937g-1.6 meq-4 fold excess-0.1M) and DIC (0.25 ml-1.6 meq-4 fold excess-0.1M) in DCM/DMF (50/50). After washes the resin was stirred with Piperidine 20% in DMF (5 min and 20 min) in order to remove the Fmoc group. After washes, this same cycle was repeated with Fmoc-Asp(OtBu), Fmoc(^{15}N-2-^{13}C)Gly (2 fold excess only), Fmoc-Lys(Boc), Fmoc-Leu, Fmoc-Thr(OtBu), Fmoc-Asn and Boc-Cys(Acm). After the last coupling, the Boc group was left on the peptide. The resin was washed thoroughly with DCM and dried under a nitrogen stream. Yield was 1.49g (Expected: ~1.7g).

The next step 44 was cyclization of the Boc-Cys-Asn-Thr(OtBu)-Leu-Lys(Boc)-Gly-Asp(OtBu)-Cys-Gly-mBHA resin. 600 mg of protected peptide resin were sealed in a polypropylene mesh packet. The bag was shaken in a mixture of solvent (DCM/Methanol/Water-640/280/47) in order to swell the resin. The bag was then shaken 20 min in 100 ml of a solution of iodine in the same mixture of solvent (0.4 mg I_2/ml solvent mixture). This operation was performed 4 times. No d coloration was observed after the third time. The resin was then thoroughly washed with DCM, DMF, DCM, and methanol successively.

The last step 45 was side-chain deprotection of the Cys-Asn-Thr-Leu-Lys-Gly-Asp-Cys-Gly-mBHA resin. After cyclization the resin in the polypropylene bag was reacted 1.5 hour with 100 ml of a mixture TFA/p-Cresol-Water (95/2.5/2.5). After washes with DCM and Methanol, the resin was dried 48 hours under vacuum. Yield was 560 mg.

The resulting peptide resin was analyzed for its purity and the presence of the disulfide bridge. 40 mg of resin were sealed in a propylene mesh packet and treated with HF at 0 C for 1 hour in presence of anisole (HF/Anisole: 90/10). The scavenger and by-products were extracted from the resin with cold ethyl ether. The peptide was extracted with 10% Acetic Acid and lyophilized 36 hours. The dry isolated peptide was characterized by PDMS (mass spectrography) and HPLC (high performance liquid chromatography). This analysis demonstrated that greater than 95% of the product peptide was of the correct amino acid composition, having a disulfide loop and without inter-molecular disulfide dimers.

REDOR measurements were made on the peptide resin prepared by this method, and as a control, also on dried (^{15}N-2-^{13}C) labeled glycine. The preferred REDOR methods and parameters, as previously detailed, were used. Fig. 6 illustrates the ^{15}N resonance spectral signals obtained. Signal 70 is the signal produced by dried glycine after no rotor periods. Signals 71, 72, 73 are glycine signals after 2, 4, and 8 rotor periods, respectively. Signals 74, 75, 76, and 77 are the peptide resin signals after 0, 2, 4, and 8 rotor periods, respectively.

Fig. 7 illustrates the data analysis. As in Fig. 5, axis 81 is the $\Delta S/S$ axis, and axis 82 is the λ axis. The variables are as used in equation 5. Graph 83 is defined by equation 5, and is the initial rising part of the full curve shown in Fig. 5. Data points 84, 85, 86, and 87 are best fits of the data for 0, 2, 4, and 8 rotor periods, respectively. At these points, the circles represent the glycine values and the squares the peptide resin values. These values correspond to a C-N distance in glycine and the peptide of 1.55 Å (and a D_{CN} of 800 Hz). Repeated measurements gave a C-N distance of 1.50 Å (and a D_{CN} of 875 Hz). The accepted distance in glycine is 1.48 Å. The above procedure was repeated for (^{15}N-1-^{13}C) labeled glycine in Cys-Asn-Thr-Leu-Lys-(^{15}N-1-^{13}C)Gly-Asp-Cys-Gly-mBHA resin, and the measured C-N distance of 2.50 Å is in excellent agreement with the predicted value of 2.46 Å.

Thus REDOR accuracy to better that 0.1 Å is demonstrated. Also demonstrated is the peptide resin as an appropriate substrate for NMR measurements. Inter-molecular dipole-dipole interactions between adjacent peptides did not interfere. Also the overlap of the distances measured in free glycine and in glycine incorporated in the peptide demonstrated that the peptide was held sufficiently rigidly by the resin that any remaining peptide motions did not interfere with the NMR measurements.

7. SPECIFIC EMBODIMENTS, CITATION OF REFERENCES

The present invention is not to be limited in scope by the specific embodiments described herein. Indeed, various modifications of the invention in addition to those described herein will become apparent to those skilled in the art from the foregoing description and accompanying figures. Such modifications are intended to fall within the scope of the appended claims.

Various publications are cited herein, the disclosures of which are incorporated by reference in their entireties.

8. COMPUTER PROGRAM LISTINGS

These computer program listings are copyright 1995 of CuraG n, Inc. © 1995 CuraGen, Inc.

```
*****************************************************************
****************************************************************
                       START OF LISTING
*****************************************************************
*****************************************************************

*****************************************************************
*****************************************************************
                       C CODE ROUTINES
*****************************************************************
*****************************************************************

*****************************************************************
                    MAKEFILE AND GO PROC
*****************************************************************

MAKEFILE:
OPTIONS=-mips2 -ansi -g -fullwarn -O0
peptide.ex: random.o peptide.o peptide1.o peptide2.o peptide3.o
peptide4.o \
           peptide5.o peptide6.o peptide7.o
     cc $(OPTIONS) random.o peptide*.o -lm -o peptide.ex
random.o: random.c
     cc $(OPTIONS) -c random.c
peptide.o: peptide.c *.h
     cc $(OPTIONS) -c peptide.c
peptide1.o: peptide1.c *.h
     cc $(OPTIONS) -c peptide1.c
peptide2.o: peptide2.c *.h
     cc $(OPTIONS) -c peptide2.c
peptide3.o: peptide3.c *.h
     cc $(OPTIONS) -c peptide3.c
peptide4.o: p ptid 4.c *.h
     cc $(OPTIONS) -c peptide4.c
```

```
peptide5.o: peptide5.c *.h
     cc $(OPTIONS) -c peptide5.c
peptide6.o: peptide6.c *.h
     cc $(OPTIONS) -c peptide6.c
peptide7.o: peptide7.c *.h
     cc $(OPTIONS) -c peptide7.c

GO PROC:
peptide.ex << EOF
0.1
1
CGGGGGGC
EOF

******************************************************************
                    MAIN PROGRAM - PEPTIDE.C
******************************************************************

#define MAIN
#include "peptide.h"
/* The main program stub */
void main(int argc, char *argv[], char *envp[])
{
  logical *cyclic;
  int n_peptides, max_atoms_per_unit;
  int *n_amino_acids, *n_atoms_total, *n_side, *n_main;
  rigid_unit **peptide;
  torsion_list **torsion;
  hbond_list **hbond;
  atom_list **atom, **atom2;
  atom_info **atom_tmp;
  vector *twig[KMAX];
  int ***bond_table;
  string *sequence;
  int i, j;
  int list_num, max_atoms_total;
  double seed;
  regrowth **main, **side;
  printf("Enter random number seed ");
  scanf("%lf", &seed);
  ran2(seed);
/* get linear sequences */
  get_sequence(&sequence, &n_peptides);
  printf("\n");
/* allocate memory for arrays */
  if ((peptide = (rigid_unit **)
     malloc(n_peptides*sizeof(rigid_unit *))) == NULL)
    out_of_memory();
  if ((torsion = (torsion_list **)
     malloc(n_peptides*sizeof(torsion_list *))) == NULL)
    out_of_memory();
  if ((hbond = (hbond_list **) malloc(n_peptides*sizeof(hbond_list
*)))==NULL)
    out_of_memory();
  if ((atom = (atom_list **) malloc(n_peptides*sizeof(atom_list
*))) == NULL)
    out_of_memory();
  if ((atom2 = (atom_list **) malloc(n_peptides*sizeof(atom_list
*))) == NULL)
    out_of_memory();
  if ((atom_tmp = (atom_info **) malloc(n_peptides*sizeof(atom_info
*)))
      == NULL) out_of_memory();
  if ((main = (regrowth **) malloc(n_peptides*sizeof(regrowth *)))
== NULL)
    out_of_memory();
  if ((side = (regrowth **) malloc(n_peptides*sizeof(regrowth *)))
== NULL)
    out_of_memory();
  if ((bond_table = (int ***) malloc(n_peptides*sizeof(int **)))
== NULL)
    out_of_memory();
  if ((n_amino_acids = (int *) malloc(n_peptides*sizeof(int))) ==
NULL)
    out_of_memory();
  if ((n_atoms_total = (int *) malloc(n_peptides*sizeof(int))) ==
NULL)
    out_of_memory();
  if ((cyclic = (logical *) malloc(n_peptides*sizeof(logical))) ==
NULL)
    out_of_m mory();
  if ((n_main = (int *) malloc(n_peptides*sizeof(int))) == NULL)
    out_of_memory();
  if ((n_side = (int *) malloc(n_peptides*sizeof(int))) == NULL)
    out_of_memory();
  for(i=0; i<n_peptides; i++) {
    n_amino_acids[i] = (int) strlen(sequence[i]);
  }
/* read in parameter files */
  read_torsion_data();
  read_lj_data();
  read_hbond_data();
  max_atoms_per_unit = 0;
/* read in geometric sequence information */
  max_atoms_total = 0;
  for (i=0; i<n_peptides; i++) {
    peptide[i] = read_peptide_data(sequence[i], &n_atoms_total[i],
                                   &max_atoms_per_unit);
    cyclic[i] = (n_amino_acids[i] > 1) && (sequence[i][0] == 'C')
&&
                (sequence[i][n_amino_acids[i]-1]=='C');
    if (cyclic[i]) peptide[i] = modify_cystine_ends(peptide[i],
                                         n_amino_acids[i],
                                         &n_atoms_total[i]);
    if   (n_atoms_total[i]>max_atoms_total)   max_atoms_total   =
n_atoms_total[i];
    n_main[i]   =   (cyclic[i])   ?   2*n_amino_acids[i]   +   3   :
2*n_amino_acids[i] + 1;
    n_side[i] = n_amino_acids[i];
  }
/* allocate sub arrays */
  for (i=0; i<KMAX; i++)
       i f   ( ( t w i g [ i ]   =   ( v e c t o r   * )
malloc(max_atoms_total*sizeof(vector)))
       == NULL) out_of_memory();
  for(i=0; i<n_peptides; i++) {
      i f   ( ( a t o m [ i ]   =   ( a t o m _ l i s t   * )
malloc(n_atoms_total[i]*sizeof(atom_list)))
         == NULL) out_of_memory();
        if  ((atom2[i]  =  (atom_list  *)
malloc(n_atoms_total[i]*sizeof(atom_list)))
         == NULL) out_of_memory();
        if  ((atom_tmp[i]=(atom_info  *)
malloc(n_atoms_total[i]*sizeof(atom_info)))
         == NULL) out_of_memory();
    if ((main[i] = (regrowth *)
       malloc(n_main[i]*sizeof(regrowth))) == NULL)
      out_of_memory();
    if ((side[i] = (regrowth *)
       malloc(n_side[i]*sizeof(regrowth))) == NULL)
      out_of_memory();
        if  ((bond_table[i]  =  (int  **)
malloc(n_atoms_total[i]*sizeof(int *)))
         == NULL) out_of_memory();
    for (j=0; j<n_atoms_total[i]; j++)
          if  ((bond_table[i][j]  =  (int  *)
malloc(MAX_BONDS*sizeof(int)))
         == NULL) out_of_memory();
  }
/* loop over all peptides */
  for (i=0; i<n_peptides; i++) {
    get_main_side(peptide[i],  main[i],  side[i],  &n_main[i],
&n_side[i]);
/* determine connections */
    initialize_connection_table(bond_table[i], n_atoms_total[i]);
    list_num = 0;
    make_connection_table(bond_table[i],  &list_num,  peptide[i],
peptide[i]);
    /*print_connection_table(bond_table[i], n_atoms_total[i]);*/
    list_num = 0;
/* assign noncoordinate information in atom array */
    assign_atom_pointers(&list_num,   peptide[i],   peptide[i],
atom[i]);
/* g t H-bonds and torsion lists */
    get_hbonds(&hbond[i], atom[i], n_atoms_total[i]);
    /*print_hbonds(hbond[i], atom[i]);*/
    list_num = 0;
    torsion[i] = NULL;
    get_torsions(&torsion[i], bond_table[i], &list_num, atom[i],
p ptide[i],
                    peptide[i]);
    assign_lj_parameters(peptide[i], peptide[i]);
/* copy noncoordinate information in atom to atom2 */
    for (j=0; j<n_atoms_total[i]; j++) atom2[i][j] = atom[i][j];
  }
/* do the Monte Carlo */
  do_mc(peptide[0],  torsion[0],  hbond[0],  atom[0],  atom2[0],
atom_tmp[0],
        twig, main[0], side[0], n_amino_acids[0],
        n_atoms_total[0], n_main[0], n_side[0], cyclic[0]);
  /*print_torsions(torsion[0], atom[0]);*/
  write_car_file(n_amino_acids[0],  n_atoms_total[0],  atom[0],
"test.car");
}
#undef MAIN

*****************************************************************
               INPUT/OUTPUT ROUTINES - PEPTIDE1.C
*****************************************************************

               /* input/output routines */
#include "peptide.h"
/* hardcoded AMBER rules have the keyword AMBER nearby
*/
#define NT_CT_DISTANCE 1.4750
#define S_S_DISTANCE 2.0380
#define P_CHARGE 0.048
#define C_CHARGE1 -0.098
#define C_CHARGE2 0.050
#define C_CHARGE3 0.050
#define C_CHARGE4 0.824
#define C_CHARGE5 -0.405
#define C_CHARGE6 -0.405
/* This function is called when out of memory
*/
```

```
void out_of_memory(void)
{
  printf("Out of memory error\n");
  exit(1);
}
/* This routine returns the 1-letter amino acide sequences
*/
void get_sequence(string **sequence, int *n_peptides)
{
#define SEQUENCE_LENGTH 80
  int i;
  printf("Enter number of peptides: ");
  scanf("%d", n_peptides);
  if ((*sequence = (string *) malloc(*n_peptides*sizeof(string)))
== NULL)
    out_of_memory();
  for (i=0; i<*n_peptides; i++)
       if   (((*sequence)[i]   =   (string)
malloc(SEQUENCE_LENGTH*sizeof(char)))
         == NULL) out_of_memory();
  for (i=0; i<*n_peptides; i++) {
    printf("Enter peptide sequence %d: ",i);
    scanf("%s", (*sequence)[i]);
  }
#undef SEQUENCE_LENGTH
}
/* read in the data files associated with this sequence
*/
rigid_unit *read_peptide_data(string sequence, int *n_atoms_total,
                              int *max_atoms_per_unit)
{
  int i, n_amino_acids;
  char name[]="?.dat";
  acid_label label;
  rigid_unit *u1, *u2, *ret;

/* check amino acids in sequence */
  n_amino_acids = strlen(sequence);
  for(i=0; i<n_amino_acids; i++) {
      label = amino_acid_code(sequence[i]);
      if (label == BAD) {
        printf("Invalid amino acid code %c\n", sequence[i]);
        exit(1);
        }
      if (label == P) {
        printf("Proline not yet supported\n");
        exit(1);
      }
    }
    *n_atoms_total = 0;
  /* add unit A */
    label = amino_acid_code(sequence[0]);
    u1   =   read_unit("unitA.dat",   label,   0,   n_atoms_total,
  max_atoms_per_unit);
    ret = u1;
    for(i=0; i<n_amino_acids; i++) {
      name[0] = sequence[i];
      label = amino_acid_code(sequence[i]);
  /* add unit B */
      u2   =   read_unit("unitB.dat",   label,   i,   n_atoms_total,
  max_atoms_per_unit);
      u2->type = nonCunit;
  /* follow IUPAC naming rules if glycine */
      if (label == G) strcpy(u2->atom[1].name, "HA1");
  /* follow AMBER charge rules if alanine or proline */
      if (label == A || label == P) u2->atom[1].charge = P_CHARGE;
      if   (i==0)   u2->head.axis   =   vector_scale(u2->head.axis,
  NT_CT_DISTANCE);
      couple_unit(u1,u2);
      u1 = u2;
  /* add residue */
      u2    =    read_unit(name,    label,    i,    n_atoms_total,
  max_atoms_per_unit);
      couple_unit(u1, u2);
  /* add unit C or D */
      u2   =   read_unit((i==n_amino_acids-1)   ?   "unitD.dat"   :
  "unitC.dat",
        label, i, n_atoms_total, max_atoms_per_unit);
    if (i < n_amino_acids-1) {
/* align incoming and outgoing bonds */
      u2->bond[0]->tail.axis = vector_scale(u2->head.axis, 1.0);
      u2->type = Cunit;
      label = amino_acid_code(sequence[i+1]);
      u2->atom[2].residue = u2->atom[3].residue = label;
      u2->atom[2].residue_num = u2->atom[3].residue_num = i+1;
    }
    couple_unit(u1, u2);
    u1 = u2;
  }
  return(ret);
}
/* This routine reads in a rigid unit data file
*/
rigid_unit   *read_unit(string   file,   acid_label   label,   int
residue_num,
                      int *n_atoms_total, int *max_atoms_per_unit)
{
#define LINE_LEN 200
  FILE *fp;
  int i, j, k, i1, n_rigid_units;
  char stmp1[NAME_LENGTH], stmp2[NAME_LENGTH], line[LINE_LEN];
  rigid_unit **utmp;
  if ((fp = fopen(file, "r")) == NULL) {
    printf("Data file %s does not exist\n", file);
    exit(1);
  }
/* read in number of rigid units */
  getline(line, LINE_LEN, fp);
  sscanf(line, "%d", &n_rigid_units);
/*  printf("%d\n",n_rigid_units);  */
  if ((utmp = (rigid_unit **)
     malloc(n_rigid_units*sizeof(rigid_unit *))) == NULL)
     out_of_memory();
/* allocate rigid unit */
  for (i=0; i<n_rigid_units; i++) {
    if ((utmp[i] = (rigid_unit *)
       malloc(sizeof(rigid_unit))) == NULL) out_of_memory();
      utmp[i]->type = UNKNOWN;
      getline(line,LINE_LEN,fp);
      sscanf(line, "%d", &utmp[i]->n_atoms);
      *n_atoms_total += utmp[i]->n_atoms;
      if (utmp[i]->n_atoms > *max_atoms_per_unit)
        *max_atoms_per_unit = utmp[i]->n_atoms;
/*      printf("%d\n",utmp[i]->n_atoms); */
      if ((utmp[i]->atom = (atom_info *)
         malloc(utmp[i]->n_atoms*sizeof(atom_info))) == NULL)
         out_of_memory();
/* read in atoms */
      for(j=0; j<utmp[i]->n_atoms; j++) {
        getline(line,LINE_LEN,fp);
        sscanf(line, "%s %lf %lf %lf %s %d %s %s %lf",
               utmp[i]->atom[j].name,
               &utmp[i]->atom[j].position.x,
               &utmp[i]->atom[j].position.y,
               &utmp[i]->atom[j].position.z,
               &stmp1, &i1,
               utmp[i]->atom[j].type, &stmp2,
               &utmp[i]->atom[j].charge);
/*        printf("%s %lf %lf %lf %s %lf\n",
               utmp[i]->atom[j].name,
               utmp[i]->atom[j].position.x,
               utmp[i]->atom[j].position.y,
               utmp[i]->atom[j].position.z,
               utmp[i]->atom[j].type,
               utmp[i]->atom[j].charge); */
        utmp[i]->atom[j].residue = label;
        utmp[i]->atom[j].residue_num = residue_num;
      }
    }
    for (i=0; i<n_rigid_units; i++) {
/* allocate incoming bond vector information */
      getline(line,LINE_LEN,fp);
      sscanf(line, "%d %d %d %d %d", &i1, &utmp[i]->head.bond[0],
           &utmp[i]->head.bond[1], &utmp[i]->head.bond[2],
           &utmp[i]->head.bond[3]);
/*      printf("%d %d %d %d %d\n",i1, utmp[i]->h ad.bond[0],
             utmp[i]->head.bond[1], utmp[i]->head.bond[2],
             utmp[i]->head.bond[3]); */
      for (j=4; j<MAX_BONDS; j++) utmp[i]->head.bond[j] = -1;
      utmp[i]->head.atom_num = i1;
      getline(line,LINE_LEN,fp);
      sscanf(line, "%lf %lf %lf", &utmp[i]->head.axis.x,
                                  &utmp[i]->head.axis.y,
                                  &utmp[i]->head.axis.z);
/*     printf("%lf %lf %lf\n",utmp[i]->head.axis.x,
                                  utmp[i]->head.axis.y,
                                  utmp[i]->head.axis.z); */

utmp[i]->head.axis.x=utmp[i]->atom[i1].position.x-utmp[i]->head.
axis.x;

utmp[i]->head.axis.y=utmp[i]->atom[i1].position.y-utmp[i]->head.
axis.y;

utmp[i]->head.axis.z=utmp[i]->atom[i1].position.z-utmp[i]->head.
axis.z;
/* allocate outgoing bond pointers */
     getline(line,LINE_LEN,fp);
     sscanf(line, "%d", &utmp[i]->n_bonds);
     if ((utmp[i]->bond = (bond_type **)
        malloc(utmp[i]->n_bonds*sizeof(bond_type *))) == NULL)
        out_of_memory();
     for (j=0; j<utmp[i]->n_bonds; j++) {
       if ((utmp[i]->bond[j] = (bond_type *)
          malloc(sizeof(bond_type))) == NULL)
          out_of_memory();
       getline(line,LINE_LEN,fp);
       sscanf(line, "%d", &i1);
/*       printf("%d\n",i1); */
       utmp[i]->bond[j]->next = (i1==-1) ? NULL : utmp[i1];
       getline(line,LINE_LEN,fp);
        sscanf(line,    "%d    %d    %d    %d    %d",    &i1,
&utmp[i]->bond[j]->tail.bond[0],
                                   &utmp[i]->bond[j]->tail.bond[1],
                                   &utmp[i]->bond[j]->tail.bond[2],
                              &utmp[i]->bond[j]->tail.bond[3]);
 /*                    printf("%d   %d   %d   %d   %d\n",   i1,
 utmp[i]->bond[j]->tail.bond[0],
                              utmp[i]->bond[j]->tail.bond[1],
                              utmp[i]->bond[j]->tail.bond[2],
                              utmp[i]->bond[j]->tail.bond[3]);*/
      for (k=4; k<MAX_BONDS; k++) utmp[i]->bond[j]->tail.bond[k]
 = -1;
      utmp[i]->bond[j]->tail.atom_num= i1;
      getline(line,LINE_LEN,fp);
      sscanf(line, "%lf %lf %lf", &utmp[i]->bond[j]->tail.axis.x,
                              &utmp[i]->bond[j]->tail.axis.y,
                              &utmp[i]->bond[j]->tail.axis.z);
         u t m p [ i ] - > b o n d [ j ] - > t a i l . a x i s . x   - =
 utmp[i]->atom[i1].position.x;
         u t m p [ i ] - > b o n d [ j ] - > t a i l . a x i s . y   - =
 utmp[i]->atom[i1].position.y;
         u t m p [ i ] - > b o n d [ j ] - > t a i l . a x i s . z   - =
 utmp[i]->atom[i1].position.z;
      utmp[i]->bond[j]->tail.axis =
              vector_scale(utmp[i]->bond[j]->tail.axis,1.0);
    }
  }
  fclose(fp);
  return(utmp[0]);
#undef LINE_LEN
}
/* This routine couples two rigid units
*/
void couple_unit(rigid_unit *unit1, rigid_unit *unit2)
{
  bond_type **bond;
  for(bond=unit1->bond; bond[0]->next; bond++) ;
  bond[0]->next = unit2;
}
/* This routine turns a linear CX_nC peptide into a cyclic
   disulfide-bonded peptide
*/
rigid_unit   *modify_cystine_ends(rigid_unit   *unit,   int
n_amino_acids,
                                   int *n_atoms_total)
{
  int i;
  rigid_unit *unit1, *unit2, *unit3, *unit4, *unit5, *unit6;
  double len;
  vector head1, head2;
  bond_type *btmp;
/* get new first unit */
  unit1 = unit->bond[0]->next;
  unit2 = unit1->bond[0]->next;
  unit3 = unit2->bond[0]->next;
/* save head vectors */
  head1 = unit1->head.axis;
  head2 = unit2->head.axis;
/* modify A unit to be a side group */
  len = vector_length(unit1->head.axis);
  unit->head = unit->bond[0]->tail;
  unit->head.axis.x *= -len;
  unit->head.axis.y *= -len;
  unit->head.axis.z *= -len;
  unit->n_bonds = 0;
/* modify C_alpha head */
   len = vector_length(unit2->head.axis);
   unit1->head = unit1->bond[0]->tail;
   unit1->head.axis.x *= -len;
   unit1->head.axis.y *= -len;
   unit1->head.axis.z *= -len;
/* modify C_beta head */
   len = vector_length(unit3->head.axis);
   unit2->head = unit2->bond[0]->tail;
   unit2->head.axis.x *= -len;
   unit2->head.axis.y *= -len;
   unit2->head.axis.z *= -len;
/* modify S tail */
   unit3->bond = unit->bond;
   unit3->head.bond[2] = -1;
   unit3->bond[0]->tail = unit3->head;
   unit3->bond[0]->tail.axis   =   vector_scale(unit3->head.axis,
 -1.0);
    unit3->bond[0]->next = unit2;
    unit3->n_bonds = 1;
    unit3->n_atoms--;
    (*n_atoms_total)--;
 /* modify S head */
    unit3->head.axis = unit3->atom[0].position;
    unit3->head.axis.x -= unit3->atom[3].position.x;
    unit3->head.axis.y -= unit3->atom[3].position.y;
    unit3->head.axis.z -= unit3->atom[3].position.z;
 /* modify C_beta tail */
    unit2->bond[0]->tail.axis = vector_scale(head2, -1.0);
    unit2->bond[0]->next = unit1;
 /* modify C_alpha tail */
    unit1->bond[0]->tail.axis = vector_scale(head1, -1.0);
    unit1->bond[0]->next = unit;
    unit4 = unit1;
 /* find last B unit */
   for (i=1; i<n_amino_acids; i++) {
     unit4 = unit4->bond[unit4->n_bonds-1]->next;
     unit4 = unit4->bond[unit4->n_bonds-1]->next;
   }
   unit5 = unit4->bond[0]->next;
   unit6 = unit5->bond[0]->next;
 /* swap bond 0 and bond1 for unit 4*/
   btmp = unit4->bond[0];
   unit4->bond[0] = unit4->bond[1];
   unit4->bond[1] = btmp;
 /* modify S tail */
    if ((unit6->bond = (bond_type **) malloc(sizeof(bond_type *)))
== NULL)
       out_of_memory();
    if ((unit6->bond[0] = (bond_type *) malloc(sizeof(bond_type)))
== NULL)
       out_of_memory();
   unit6->head.bond[2] = -1;
   unit6->bond[0]->tail = unit6->h ad;
   unit6->bond[0]->next = unit3;
   unit6->n_bonds = 1;
```

```
    unit6->n_atoms--;
    (*n_atoms_total)--;
    unit6->bond[0]->tail.axis = unit6->atom[3].position;
    unit6->bond[0]->tail.axis.x -= unit6->atom[0].position.x;
    unit6->bond[0]->tail.axis.y -= unit6->atom[0].position.y;
    unit6->bond[0]->tail.axis.z -= unit6->atom[0].position.z;
      unit6->bond[0]->tail.axis =
vector_scale(unit6->bond[0]->tail.axis, 1.0);
/* use AMBER S-S bond length */
    unit3->head.axis = vector_scale(unit3->head.axis, S_S_DISTANCE);
/* modify cystine S types to obey AMBER rules */
    strcpy(unit3->atom[0].type, "S");
    strcpy(unit6->atom[0].type, "S");
/* modify cystine charges to obey AMBER rules */
    unit2->atom[0].charge = C_CHARGE1;
    unit2->atom[1].charge = C_CHARGE2;
    unit2->atom[2].charge = C_CHARGE3;
    unit3->atom[0].charge = C_CHARGE4;
    unit3->atom[1].charge = C_CHARGE5;
    unit3->atom[2].charge = C_CHARGE6;
    unit5->atom[0].charge = C_CHARGE1;
    unit5->atom[1].charge = C_CHARGE2;
    unit5->atom[2].charge = C_CHARGE3;
    unit6->atom[0].charge = C_CHARGE4;
    unit6->atom[1].charge = C_CHARGE5;
    unit6->atom[2].charge = C_CHARGE6;
/* reassign first unit */
  return(unit3);
}
/* This routine determines the main and side unit pointers
*/
void get_main_side(rigid_unit *unit, regrowth *main, regrowth
*side,
                  int *n_main, int *n_side)
{
  rigid_unit *start, *unit2, *lastmain;
  regrowth *main0;
  int i;
  main0 = main;
```

```
    *n_side = 0;
    *n_main = 0;
    start = unit;
    lastmain = NULL;
    do {
      main->unit = unit;
      main->prev = lastmain;
      main++;
      (*n_main)++;
      for (i=0; i<unit->n_bonds-1; i++) {
        unit2 = unit->bond[i]->next;
        if (unit2->atom[0].residue != G) {
          side->unit = unit2;
          side->prev = unit;
          side++;
          (*n_side)++;
        }
      }
      lastmain = unit;
      unit = unit->bond[i]->next;
    } while (start != unit && unit->n_bonds > 0);
    if (unit->n_bonds == 0) {
      main->unit = unit;
      main->prev = lastmain;
      main++;
      (*n_main)++;
    } else {
      main0->prev = lastmain;
    }
}
/* This routine reads in the torsion data file
*/
void read_torsion_data(void)
{
#define LINE_LEN 200
  FILE *fp;
  char line[LINE_LEN];
  int n_torsions, itmp, i;
  double ftmp;
```

```
   torsion_data **data;
   if ((fp = fopen("torsion.dat", "r")) == NULL) {
     printf("Data file torsion.dat does not exist\n");
     exit(1);
   }
   getline(line, LINE_LEN, fp);
   sscanf(line, "%d", &n_torsions);
   if ((torsion_data_list = (torsion_data **)
      malloc((n_torsions+1)*sizeof(torsion_data  *)))   ==   NULL)
 out_of_memory();
   data = torsion_data_list;
   data[n_torsions] = NULL;
   for (i=0; i<n_torsions; i++) {
     if ((data[i] = (torsion_data *) malloc(sizeof(torsion_data)))
== NULL)
       out_of_memory();
     getline(line, LINE_LEN, fp);
     sscanf(line, "%lf %d %s %s %s %s %lf %lf %lf %lf %lf %lf",
            &ftmp, &itmp, data[i]->type1,
             data[i]->type2, data[i]->type3, data[i]->type4,
             &data[i]->v0[0], &data[i]->phi0[0],
             &data[i]->v0[1], &data[i]->phi0[1],
             &data[i]->v0[2], &data[i]->phi0[2]);
     data[i]->phi0[0] *= PI/180.0;
     data[i]->phi0[1] *= PI/180.0;
     data[i]->phi0[2] *= PI/180.0;
   }
   fclose(fp);
#undef LINE_LEN
}
/* This routine reads in the Lennard-Jones data file
*/
void read_lj_data(void)
{
#define LINE_LEN 200
  FILE *fp;
  char line[LINE_LEN];
  int n_terms, itmp, i;
  double ftmp;
  lj_data **data;
  if ((fp = fopen("lj_param.dat", "r")) == NULL) {
    printf("Data file lj_param.dat does not exist\n");
    exit(1);
  }
  getline(line, LINE_LEN, fp);
  sscanf(line, "%d", &n_terms);
   if  ((lj_data_list  =  (lj_data  **)
malloc((n_terms+1)*sizeof(lj_data *)))
       == NULL) out_of_memory();
  data = lj_data_list;
  data[n_terms] = NULL;
  for (i=0; i<n_terms; i++) {
    if ((data[i] = (lj_data *) malloc(sizeof(lj_data))) == NULL)
      out_of_memory();
    getline(line, LINE_LEN, fp);
    sscanf(line, "%lf %d %s %lf %lf", &ftmp, &itmp, data[i]->type,
                      &data[i]->ri, &data[i]->ei);
  }
  fclose(fp);
#undef LINE_LEN
}
/* This routine reads in the H-bond data file
*/
void read_hbond_data(void)
{
#define LINE_LEN 200
  FILE *fp;
  char line[LINE_LEN];
  int n_terms, itmp, i;
  double ftmp;
  hbond_data **data;
  if ((fp = fopen("hbond.dat", "r")) == NULL) {
    printf("Data file hbond.dat does not exist\n");
    exit(1);
  }
  getline(line, LINE_LEN, fp);
  sscanf(line, "%d", &n_terms);
  if ((hbond_data_list = (hbond_data **)
      malloc((n_terms+1)*sizeof(hbond_data    *)))    ==    NULL)
out_of_memory();
  data = hbond_data_list;
  data[n_terms] = NULL;
  for (i=0; i<n_terms; i++) {
    if ((data[i] = (hbond_data *) malloc(sizeof(hbond_data))) ==
NULL)
      out_of_memory();
    getline(line, LINE_LEN, fp);
    sscanf(line, "%lf %d %s %s %lf %lf",
       &ftmp, &itmp, data[i]->type1,
       data[i]->type2, &data[i]->a, &data[i]->b);
  }
  fclose(fp);
#undef LINE_LEN
}
/* write out the BIOSYM car files associated with this sequence
*/
void write_car_file(int n_amino_acids, int n_atoms_total, atom_list
*atom,
                    string file)
{
  int i;
  char name[NAME_LENGTH];
  FILE *fp;
  time_t t;
  if ((fp = fopen(file, "w")) == NULL) {
    printf("Cannot open car file %s\n", file);
    exit(1);
  }
  fprintf(fp, "!BIOSYM archive 3\n");
  fprintf(fp, "PBC=OFF\n\n");
  t = time(NULL);
  fprintf(fp, "!DATE %s", ctime(&t));
  for (i=0; i<n_atoms_total; i++) {
    amino_acid_code_3(atom[i].p->residue, name);
    capitaliz (name);
    if (atom[i].p->residue_num == n_amino_acids-1)
      strcat(name,"N");
    else if (atom[i].p->residue_num == 0)
      strcat(name,"n");
    else if (atom[i].p->residue == C)
      strcat(name,"H");
    fprintf(fp, "%-5s%15.9f%15.9f%15.9f %-4s %-3d     %-2s
%2c%8.3f\n",
        atom[i].p->name,
        atom[i].position.x, atom[i].position.y,
        atom[i].position.z,    name,    atom[i].p->residue_num+1,
atom[i].p->type,
        atom[i].p->type[0], atom[i].p->charge);
  }
  fprintf(fp,"end\nend\n");
  fclose(fp);
}
/* this routine returns the next valid line from the file
*/
string getline(string line, int len, FILE *fp)
{
  string ret;
  do {
    ret=fgets(line,len,fp);
    strip(line);
  } while (ret != NULL && *line=='\x0') ;
  return(ret);
}
/* strip CR and LF from the end of a string
   also ignore everything to the right of !
*/
void strip(string string)
{
  for (; *string != '\x0' && *string != '\xA' && *string != '\xD'
         && *string != '!'; string++)
     ;
   *string = '\x0';
}
/* remove commas from string, replacing with spac s
*/
void decomma(string string)
```

```
{
  for (; *string != '\0'; string++)
    if (*string == ',') *string = ' ';
}
/* This function capitalizes a string
*/
void capitalize(string s)
{
  int o;
  o = 'a' - 'A';
  for (; *s; s++) if (*s >= 'a' && *s <= 'z') *s -= o;
}
/* This function returns the 3-letter code for the amino acid
*/
void amino_acid_code_3(acid_label label, string code_3)
{
  switch (label) {
    case G: strcpy(code_3, "Gly"); break;
    case A: strcpy(code_3, "Ala"); break;
    case V: strcpy(code_3, "Val"); break;
    case L: strcpy(code_3, "Leu"); break;
    case I: strcpy(code_3, "Ile"); break;
    case S: strcpy(code_3, "Ser"); break;
    case T: strcpy(code_3, "Thr"); break;
    case D: strcpy(code_3, "Asp"); break;
    case E: strcpy(code_3, "Glu"); break;
    case N: strcpy(code_3, "Asn"); break;
    case Q: strcpy(code_3, "Gln"); break;
    case K: strcpy(code_3, "Lys"); break;
    case H: strcpy(code_3, "His"); break;
    case R: strcpy(code_3, "Arg"); break;
    case F: strcpy(code_3, "Phe"); break;
    case Y: strcpy(code_3, "Tyr"); break;
    case W: strcpy(code_3, "Trp"); break;
    case C: strcpy(code_3, "Cys"); break;
    case M: strcpy(code_3, "Met"); break;
    case P: strcpy(code_3, "Pro"); break;
    default : strcpy(code_3, "???");
  }
```

```
}
/* This function returns the 1-letter code for the amino acid
*/
void amino_acid_cod _1(acid_label label, char code_1)
{
  switch (label) {
    case G: code_1 = 'G'; break;
    case A: code_1 = 'A'; break;
    case V: code_1 = 'V'; break;
    case L: code_1 = 'L'; break;
    case I: code_1 = 'I'; break;
    case S: code_1 = 'S'; break;
    case T: code_1 = 'T'; break;
    case D: code_1 = 'D'; break;
    case E: code_1 = 'E'; break;
    case N: code_1 = 'N'; break;
    case Q: code_1 = 'Q'; break;
    case K: code_1 = 'K'; break;
    case H: code_1 = 'H'; break;
    case R: code_1 = 'R'; break;
    case F: code_1 = 'F'; break;
    case Y: code_1 = 'Y'; break;
    case W: code_1 = 'W'; break;
    case C: code_1 = 'C'; break;
    case M: code_1 = 'M'; break;
    case P: code_1 = 'P'; break;
    default : code_1 = '?';
  }
}
/* This function returns the acid label from the 1-letter amino
acid code
*/
acid_label amino_acid_code(char code_1)
{
  acid_label ret;
  switch (code_1) {
    case 'G': ret = G; break;
    case 'A': ret = A; break;
    case 'V': ret = V; break;
    case 'L': ret = L; break;
    case 'I': ret = I; break;
    case 'S': ret = S; break;
    case 'T': ret = T; break;
    case 'D': ret = D; break;
    case 'E': ret = E; break;
    case 'N': ret = N; break;
    case 'Q': ret = Q; break;
    case 'K': ret = K; break;
    case 'H': ret = H; break;
    case 'R': ret = R; break;
    case 'F': ret = F; break;
    case 'Y': ret = Y; break;
    case 'W': ret = W; break;
    case 'C': ret = C; break;
    case 'M': ret = M; break;
    case 'P': ret = P; break;
    default  : ret = BAD;
  }
  return(ret);
}
```

```
*****************************************************************
            MOLECULAR TOPOLOGY CREATION - PEPTIDE2.C
*****************************************************************
```

```
/*                      The topology creation routines
*/
#include "peptide.h"
/* This routine initializes the bond connection table
*/
void   initialize_connection_table(int   **bond_table,   int
n_atoms_total)
{
  int i,j;
  for(i=0; i<n_atoms_total; i++)
    for(j=0; j<MAX_BONDS; j++)
      bond_table[i][j] = -1;
}
```

```
/* This routine creates a connection table
*/
void make_connection_table(int **bond_table, int *table_num,
                           rigid_unit *unit, rigid_unit *start)
{
  int i, *j, i1, save[MAX_BONDS];
  i1 = unit->head.atom_num + *table_num;
  for (j=unit->head.bond; *j != -1; j++) {
    add_connection(bond_table, i1, *j+*table_num);
    add_connection(bond_table, *j+*table_num, i1);
  }
  for (i=0; i<unit->n_bonds; i++) {
    i1 = unit->bond[i]->tail.atom_num + *table_num;
    for (j=unit->bond[i]->tail.bond; *j != -1; j++) {
      add_connection(bond_table, i1, *j+*table_num);
      add_connection(bond_table, *j+*table_num, i1);
    }
    save[i] = unit->bond[i]->tail.atom_num + *table_num;
  }
  *table_num += unit->n_atoms;
  for (i=0; i<unit->n_bonds; i++) {
    i1 = unit->bond[i]->next->head.atom_num;
    if (unit->bond[i]->next != start) i1 += *table_num;
    add_connection(bond_table, save[i], i1);
    add_connection(bond_table, i1, save[i]);
    if (unit->bond[i]->next != start)
       make_connection_table(bond_table,   table_num,
unit->bond[i]->next,start);
  }
}
/* This routine adds a connection to the connection table
*/
void add_connection(int **bond_table, int i1, int i2)
{
  int *i, *j;
  for (i=bond_table[i1]; *i != -1; i++) ;
  for (j=bond_table[i1]; j<i; j++) if (*j == i2) return;
  *i = i2;
}
```

```
/* This routine prints out the connection table
*/
void print_connection_table(int **bond_table, int n_atoms_total)
{
  int i, j;
  for (i=0; i<n_atoms_total; i++)  {
    printf("%5d     ",i);
    for (j=0; j<MAX_BONDS; j++) printf("%5d ", bond_table[i][j]);
    printf("\n");
  }
}
/* This routine determines the torsional terms
   p is set the head pointer and it returns the tail pointer
*/
void  get_torsions(torsion_list  **p,  int  **bond_table,  int
*table_num,
                  atom_list *atom, rigid_unit *unit, rigid_unit
*start)
{
  int i, save[MAX_BONDS];
  static torsion_list *q;
  static int i2, *j, *k;
  rigid_unit *new_unit;
  if (!*p) q = NULL;
  for (i=0; i<unit->n_bonds; i++)
    save[i] = unit->bond[i]->tail.atom_num + *table_num;
  *table_num += unit->n_atoms;
  for (i=0; i<unit->n_bonds; i++) {
    new_unit = unit->bond[i]->next;
    i2 = new_unit->head.atom_num;
    if (new_unit != start) i2 += *table_num;
    for (j=bond_table[save[i]]; *j != -1; j++)
      for (k=bond_table[i2]; *k != -1; k++)
        if (*j != i2 && save[i] != *k)
          if (!*p)
            *p = q = add_torsion(bond_table, atom, *j, save[i], i2,
*k);
          else
            if (q->next = add_t rsion(bond_tabl , atom, *j,
                                                    save[i], i2, *k))
              q = q->next;
     if (new_unit != start)
        get_torsions(p, bond_table, table_num, atom, new_unit,
start);
  }
}
/* This routine adds a torsion to the torsion list
   Wildcards on i and l (simultaneously) are allowed for
*/
torsion_list *add_torsion(int **bond_table, atom_list *atom, int
i, int j,
                                 int k, int l)
{
  torsion_list t, *v;
  char wild[]="*";
  int degen, itmp;
/* count degeneracy for "general" torsions--don't count the torsion
axis! */
/* "specific" torsions have a degeneracy of 1, "general" have a
degeneracy
    of degen */
  for (itmp=0; bond_table[j][itmp] != -1; itmp++) ;
  for (degen=0; bond_table[k][degen] != -1; degen++) ;
  itmp--;
  degen--;
  degen *= itmp;
  t.degen = 1;
/*  printf("%s %s %s %s %d\n",
                                 atom[i].p->name,   atom[j].p->name,
atom[k].p->name,
                              atom[l].p->name, degen);  */
  t.next = NULL;
  t.num[0] = i;
  t.num[1] = j;
  t.num[2] = k;
  t.num[3] = l;
/* "specific" torsions */
  if    (!lookup_torsion_data(atom[i].p->type,    atom[j].p->type,
atom[k].p->type,
                                 atom[l].p->type, &t.p)) {
/* "general" torsions */
      if    (!lookup_torsion_data(wild,    atom[j].p->type,
atom[k].p->type,
                                     wild,& t.p)) {
             printf("Torsional data not found for %s %s %s %s\n",
                            atom[i].p->type,    atom[j].p->type,
atom[k].p->type,
                       atom[l].p->type);
             return(NULL);
           }
      t.degen = degen;
      }
/* only report nonzero torsional terms--this will screw up the 1/2
factor
    for AMBER! */
/*     if   (t.p->v0[0]==0   &&  t.p->v0[1]==0   &&  t.p->v0[2]==0)
return(NULL); */
   if ((v = (torsion_list *)
     malloc(sizeof(torsion_list))) == NULL) out_of_memory();
   *v = t;
   return(v);
}
/* This routine looks up the parameters for a torsional term in the
    torsion data base
*/
logical  lookup_torsion_data(string  type1,  string  type2,  string
type3,
                                   string type4, torsion_data **p)
{
   torsion_data **l;
   for (l=torsion_data_list; *l; l++) {
     if   (strcmp((*l)->type1,   type1)==0   &&   strcmp((*l)->typ 2,
type2)==0 &&
                s t r c m p ( ( * l ) - > t y p e 3 , t y p e 3 ) = = 0    & &
strcmp((*l)->type4,typ 4)==0)
         goto don ;
     if   (strcmp((*l)->type1,   typ 4)==0   &&   strcmp((*l)->type2,
type3)==0 &&
              strcmp((*1)->type3,type2)==0    &&
strcmp((*1)->type4,type1)==0)
       goto done;
  }
  return(FALSE);
done: ;
  *p = *1;
  return(TRUE);
}
/* This routine prints out the torsion terms
*/
void print_torsions(torsion_list *list, atom_list *atom)
{
  torsion_list *t;
  double theta;
  for (t=list; t; t=t->next)
  {
      theta    =    torsion(atom[t->num[0]].position,
atom[t->num[1]].position,
                            atom[t->num[2]].position,
atom[t->num[3]].position);
     printf("%4-s %4-s %4-s %4-s",atom[t->num[0]].p->name,
                              atom[t->num[1]].p->name,
                              atom[t->num[2]].p->name,
                              atom[t->num[3]].p->name);
/*          printf("%4-d  %4-d  %4-d  %4-d",t->num[0],  t->num[1],
t->num[2],
                                t->num[3]); */
     printf("%4d ",t->degen);
     printf("%9.3lf %7.3lf %7.3lf %7.3lf %7.3lf %7.3lf %7.3lf\n",
            180.0*theta/PI,
            t->p->v0[0], t->p->v0[1], t->p->v0[2],
            180.0*t->p->phi0[0]/PI, 180.0*t->p->phi0[1]/PI,
            180.0*t->p->phi0[2]/PI);
  }
}
/* This routine determines the torsional angle (in radians) defined
by the
```

```
    input positions--bonded in the order p1-p2-p3-p4
*/
double torsion(vector p1, vector p2, vector p3, vector p4)
{
  vector b1, b2, b3, n1, n2;
  double dot, len, theta;
/* define bond vectors */
  b3.x = p1.x - p2.x; b3.y = p1.y - p2.y; b3.z = p1.z - p2.z;
  b2.x = p3.x - p2.x; b2.y = p3.y - p2.y; b2.z = p3.z - p2.z;
  b1.x = p4.x - p3.x; b1.y = p4.y - p3.y; b1.z = p4.z - p3.z;
  b2 = vector_scale(b2, 1.0);
  dot = vector_dot(b1,b2);
/* project bonds onto torsion axis */
  n1.x = b1.x - dot*b2.x; n1.y = b1.y - dot*b2.y; n1.z = b1.z -
dot*b2.z;
  dot = vector_dot(b3,b2);
  n2.x = b3.x - dot*b2.x; n2.y = b3.y - dot*b2.y; n2.z = b3.z -
dot*b2.z;
  len = vector_length(n1)*vector_length(n2);
  theta = vector_dot(n1,n2)/len;
/* watch out for theta=0,PI, which kill acos */
  if (theta > 1.0-EPS)
    theta = 0.0;
  else if (theta < -1.0+EPS)
    theta = PI;
  else
    theta = acos(theta);
/* get proper sign on angle */
  n1 = vector_cross(n2,n1);
  if (vector_dot(n1, b2) < 0.0) theta = -theta;
  return(theta);
}
/* This function assigns the lennard jones parameters
*/
void assign_lj_parameters(rigid_unit *unit, rigid_unit *start)
{
  int i;
  f r (i=0; i<unit->n_atoms; i++) {
    if (!lookup_lj_data(unit->atom[i].type, &unit->atom[i].ri,
            &unit->atom[i].ei)) {
        printf("Lennard-Jones parameters not found for atom %s\n",
                             unit->atom[i].type);
        exit(1);
      }
    }
    for (i=0; i<unit->n_bonds; i++)
      if (unit->bond[i]->next != start)
        assign_lj_parameters(unit->bond[i]->next, start);
}
/* This function looks up the lennard jones parameters for an atom
*/
logical lookup_lj_data(string type, double *ri, double *ei)
{
  lj_data **l;
  for (l=lj_data_list; *l; l++)
    if (strcmp((*l)->type, type)==0) {
      *ri = (*l)->ri;
      *ei = (*l)->ei;
      return(TRUE);
    }
  return(FALSE);
}
/* This routine determines the H-bonds that are in the molecule
*/
void get_hbonds(hbond_list **list, atom_list *atom, int n_atoms)
{
  int i,j;
  hbond_list t, *u, *v;
  *list = NULL;
  t.next = NULL;
  for (i=0; i<n_atoms; i++)
    for (j=i+1; j<n_atoms; j++)
      if   (lookup_hbond_data(atom[i].p->type,   atom[j].p->type,
&t.p)) {
        t.num[0] = i;
        t.num[1] = j;
        if ((v = (hbond_list *)
          malloc(sizeof(hbond_list))) == NULL) out_of_memory();
          *v = t;
          if (!*list)
            *list = u = v;
          else {
            u->next = v;
            u = u->next;
          }
        }
}
/* This function looks up the H-bond parameters for an atom pair
*/
logical lookup_hbond_data(string type1, string type2, hbond_data
**p)
{
  hbond_data **l;
  for (l=hbond_data_list; *l; l++) {
    if  (strcmp((*l)->type1,  type1)==0  &&  strcmp((*l)->type2,
type2)==0)
        goto done;
    if  (strcmp((*l)->type2,  type1)==0  &&  strcmp((*l)->type1,
type2)==0)
        goto done;
  }
  return(FALSE);
done: ;
  *p = *l;
  return(TRUE);
}
/* This function prints out the H-bonds
*/
void print_hbonds(hbond_list *l, atom_list *atom)
{
  for (; l; l=l->next) {
    printf("%s %s %lf %lf\n",
     atom[l->num[0]].p->name,  atom[l->num[1]].p->name,  l->p->a,
l->p->b);
  }
}
/* This function assigns the atom pointers
```

```
*/
void   assign_atom_pointers(int   *list_num,   rigid_unit   *unit,
rigid_unit *start,
                              atom_list *atom)
{
  int i;
  for   (i=0;   i<unit->n_atoms;   i++)   atom[i+*list_num].p   =
&unit->atom[i];
  *list_num += unit->n_atoms;
  for (i=0; i<unit->n_bonds; i++)
    if (unit->bond[i]->next != start)
      assign_atom_pointers(list_num, unit->bond[i]->next, start,
atom);
}

********************************************************************
              GEOMETRY CREATION ROUTINES - PEPTIDE3.C
********************************************************************

/*                          The geometry creation routines
*/
#include "peptide.h"
logical grow_backwards=FALSE;
/*  This  function  creates  the  Rosenbluth  factor  for  an  old
configuration
*/
void old_unit(int  *list_num,  int  n0,  int  n1,  int  n2,  double
*logrosen,
               rigid_unit *unit, rigid_unit *start, torsion_list *t,
               hbond_list  *l,  atom_list  *atom,  vector  *twig[],
vector p0,
               vector b0)
{
  int i, j;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1;
  double e;
  p1 = unit->atom[unit->h ad.atom_num].position;
  b1 = unit->head.axis;
  do_unit_sub(list_num, n0, n1, n2, logrosen, unit, t, l, atom,
twig,
              p1, b1, p0, b0, &e, p, b, FALSE);
  for (j=0; j<unit->n_bonds; j++)
    if (unit->bond[j]->next != start)
      old_unit(list_num, n0, n1, n2, logrosen, unit->bond[j]->next,
start,
              t, l, atom, twig, p[j], b[j]);
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is in one direction.
*/
void do_unit(int *list_num, int n0, int n1, int n2, double
*logrosen,
             rigid_unit *unit, rigid_unit *start, torsion_list *t,
             hbond_list *l, atom_list *atom, vector *twig[], vector
p0,
             vector b0, double *e)
{
  int i, j;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1;
  unit->list_num = *list_num;
  p1 = unit->atom[unit->head.atom_num].position;
  b1 = unit->head.axis;
  do_unit_sub(list_num, n0, n1, n2, logrosen, unit, t, l, atom,
twig,
              p1, b1, p0, b0, e, p, b, TRUE);
/* loop over remaining units */
  for (j=0; j<unit->n_bonds; j++) {
/* store side-chain regrowth info */
    if (unit->bond[j]->next != start)
      do_unit(list_num, n0, n1, n2, logrosen,
              unit->bond[j]->next, start, t, l, atom, twig, p[j],
b[j], e);
  }
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is forward.
*/
void do_backbone_f(int i, int n_main, int n_atoms_total,
                    double *logrosen,
                    regrowth *main, regrowth *side,
                    torsion_list *t, hbond_list *l,
                    atom_list *atom, vector *twig[],
                    double *e, logical new)
{
  int list_num, n1, n2;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1, p0, b0;
  if (i==0) i++;
  p0 = get_main_p0(atom, main, i);
  b0 = get_main_b0(atom, main, i);
  main += i;
  list_num = main->unit->list_num;
  n1 = n2 = n_atoms_total;
/* loop over backbone groups */
  for (; i<n_main; i++, main++) {
    p1 = main->unit->atom[main->unit->head.atom_num].position;
    b1 = main->unit->head.axis;
/* add on backbone unit */
    do_unit_sub(&list_num, 0, n1, n2, logrosen, main->unit, t, l,
atom, twig,
                p1, b1, p0, b0, e, p, b, new);
    if (!new && i < n_main-1) {
      p0 = get_main_p0(atom, main, 1);
      b0 = get_main_b0(atom, main, 1);
    } else if (new && i < n_main-1) {
      p0 = p[main->unit->n_bonds-1];
      b0 = b[main->unit->n_bonds-1];
    }
/* add on side chain */
    if (main->unit->n_bonds == 2) {
      if (new)
        do_unit(&list_num, 0, n1, n2, logrosen,
                        main->unit->bond[0]->next,
main->unit->bond[0]->next,
                t, l, atom, twig, p[0], b[0], e);
      else
        old_unit(&list_num, 0, n1, n2, logrosen,
                        main->unit->bond[0]->next,
main->unit->bond[0]->next,
                 t, l, atom, twig, p[0], b[0]);
    }
  }
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is forward.
   Side chains are rigidly rotated.
*/
void do_backbone_f_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list *atom, atom_info *atom_tmp,
                         vector *twig[],
                         double *e, logical new)
{
  int list_num, n1, n2;
  vector p[MAX_BONDS], b[MAX_BONDS], p1, b1, p1a, b1a, p0, b0;
  logical false=FALSE;

  int n_atoms, j;
  atom_info *q;
  double len;
  vector b2[MAX_BONDS], v, v2;
  if (i==0) i++;
  p0 = get_main_p0(atom, main, i);
  b0 = get_main_b0(atom, main, i);
  main += i;
  list_num = main->unit->list_num;
  n1 = n2 = n_atoms_total;
/* get first head vector */
   p1     =     atom[main->unit->list_num     +
main->unit->head.atom_num].position;
  b1 = atom[main[-1].unit->list_num +

main[-1].unit->bond[main[-1].unit->n_bonds-1]->tail.atom_num]
          .position;
  b1.x = p1.x - b1.x;
  b1.y = p1.y - b1.y;
  b1.z = p1.z - b1.z;
  for (; i<n_main; i++, main++) {
/* change unit */
    n_atoms = main->unit->n_atoms;
    q = main->unit->atom;
    if (i < n_main-1)
      main->unit->n_atoms    =    main[1].unit->list_num    -
main->unit->list_num;
    main->unit->atom = atom_tmp;
    for (j=0; j<main->unit->n_atoms; j++)
      main->unit->atom[j].position = atom[list_num+j].position;
    for (j=0; j<main->unit->n_bonds; j++) {
      b2[j] = main->unit->bond[j]->tail.axis;
      v = atom[main->unit->bond[j]->next->list_num +
               main->unit->bond[j]->next->head.atom_num].position;
      v2 = atom[main->unit->list_num +
                main->unit->bond[j]->tail.atom_num].position;
      v.x -= v2.x;
      v.y -= v2.y;
      v.z -= v2.z;
      main->unit->bond[j]->tail.axis = vector_scale(v,1.0);
    }
/* get next head vector */
    if (i < n_main-1) {
      p1a = atom[main[1].unit->list_num +
                 main[1].unit->head.atom_num].position;
      b1a = atom[main->unit->list_num +

main->unit->bond[main->unit->n_bonds-1]->tail.atom_num]
              .position;
      b1a.x = p1a.x - b1a.x;
      b1a.y = p1a.y - b1a.y;
      b1a.z = p1a.z - b1a.z;
    }
/* add on unit */
    do_unit_sub(&list_num, 0, n1, n2, logrosen, main->unit, t, 1,
atom, twig,
               p1, b1, p0, b0, e, p, b, n w);
```

```
/* change unit back */
    main->unit->n_atoms = n_atoms;
    main->unit->atom = q;
    for (j=0; j<main->unit->n_bonds; j++)
      main->unit->bond[j]->tail.axis = b2[j];
/* change head vector */
    if (!new && i < n_main-1) {
      p0 = get_main_p0(atom, main, 1);
      b0 = get_main_b0(atom, main, 1);
    } else if (new && i < n_main-1) {
      p0 = p[main->unit->n_bonds-1];
      b0 = b[main->unit->n_bonds-1];
    }
    p1 = p1a;
    b1 = b1a;
  }
}
/* This function creates the geometry of a peptide
   and the Rosenbluth factor.  The growth is backward.
*/
void do_backbone_b(int i, int n_main, int n_atoms_total,
                   double *logrosen,
                   regrowth *main, regrowth *side,
                   torsion_list *t, hbond_list *l,
                   atom_list *atom, vector *twig[],
                   double *e, logical new)
{
  int list_num, n0, n1, n2, n_bonds;
  vector p[MAX_BONDS], b[MAX_BONDS], b0, p0, tmp, p1, b1;
  if (i == n_main-1) i--;
  main += i;
  n2 = n_atoms_total;
  b0 = get_main_b0(atom, main, 1);
  for (; i>=0; i--, main--) {
    n1 = main[1].unit->list_num;
    n0 = list_num = main->unit->list_num;
/* get bond vectors */
                                 p       0                          =
atom[main[1].unit->head.atom_num+main[1].unit->list_num].position;
```

```
      b0.x = -b0.x; b0.y = -b0.y; b0.z = -b0.z;
      n_bonds = main->unit->n_bonds;
      p1 = main->unit->atom[main->unit->bond[n_bonds-1]->
                    tail.atom_num].position;
      b1 = main->unit->bond[n_bonds-1]->tail.axis;
      b1.x = -b1.x;
      b1.y = -b1.y;
      b1.z = -b1.z;
      b1 = vector_scale(b1, vector_length(main[1].unit->head.axis));
      tmp = main->unit->bond[n_bonds-1]->tail.axis;
      main->unit->bond[n_bonds-1]->tail.axis = main->unit->head.axis;
/* add on unit */
      grow_backwards = TRUE;
      do_unit_sub(&list_num, n0, n1, n2, logrosen, main->unit, t, 1,
atom, twig,
                  p1, b1, p0, b0, e, p, b, new);
      grow_backwards = FALSE;
      main->unit->bond[n_bonds-1]->tail.axis = tmp;
/* change head vector */
      if (!new && i > 0)
        b0 = get_main_b0(atom, main-1, 1);
      else if (new && i > 0)
        b0 = vector_scale(b[n_bonds-1], 1.0);
/* add on side chain */
      if (main->unit->n_bonds == 2) {
        if (new)
          do_unit(&list_num, n0, n1, n2, logrosen,
                         main->unit->bond[0]->next,
main->unit->bond[0]->next,
                  t, 1, atom, twig, p[0], b[0], e);
        else
          old_unit(&list_num, n0, n1, n2, logrosen,
                         main->unit->bond[0]->next,
main->unit->bond[0]->next,
                  t, 1, atom, twig, p[0], b[0]);
      }
   }
}
/* This function creates the geometry of a peptide
```

```
   and the Rosenbluth factor.  The growth is backward.
   Side chains are rigidly rotated.
*/
void do_backbone_b_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list  *atom,  atom_info  *atom_tmp,
vector *twig[],
                         double *e, logical new)
{
  int list_num, n0, n1, n2, n_bonds, n_atoms, j;
  vector p[MAX_BONDS], b[MAX_BONDS], b0, p0, tmp, p1, b1, p1a, b1a,
         b2[MAX_BONDS], v, v2;
  logical false=FALSE;
  atom_info *q;
  if (i == n_main-1) i--;
  main += i;
  n2 = n_atoms_total;
/* get first head unit */
  p1=atom[main->unit->bond[main->unit->n_bonds-1]->tail.atom_num
+
          main->unit->list_num].position;
   b 1    =    a t o m [ m a i n [ 1 ] . u n i t - > l i s t _ n u m    +
main[1].unit->head.atom_num].position;
  b1.x = p1.x - b1.x;
  b1.y = p1.y - b1.y;
  b1.z = p1.z - b1.z;
  b0 = get_main_b0(atom, main, 1);
  for (; i>=0; i--, main--) {
/* get current info */
    list_num = main->unit->list_num;
    n_bonds = main->unit->n_bonds;
                                   p       0                        =
atom[main[1].unit->head.atom_num+main[1].unit->list_num].position;
    b0.x = -b0.x; b0.y = -b0.y; b0.z = -b0.z;
    n1 = main[1].unit->list_num;
    n0 = list_num = main->unit->list_num;
    n_atoms = main->unit->n_atoms;
```

```
     q = main->unit->atom;
 /* change current unit */
     main->unit->n_atoms = n1 - n0;
     main->unit->atom = atom_tmp;
     for (j=0; j<main->unit->n_atoms; j++)
       main->unit->atom[j].position = atom[list_num+j].position;
 /* compute bond axes */
     for (j=0; j<n_bonds; j++) {
       b2[j] = main->unit->bond[j]->tail.axis;
       v = atom[main->unit->bond[j]->next->list_num +
                 main->unit->bond[j]->next->head.atom_num].position;
       v2 = atom[list_num +
                 main->unit->bond[j]->tail.atom_num].position;
       v.x -= v2.x;
       v.y -= v2.y;
       v.z -= v2.z;
       main->unit->bond[j]->tail.axis = vector_scale(v,1.0);
     }
     main->unit->bond[n_bonds-1]->tail.axis =
         vector_scale(get_main_b0(atom, main-i, i),
                      vector_length(main->unit->head.axis));
 /* compute new head vector */
     if (i > 0) {
                                        p    1    a              =
atom[main[-1].unit->bond[main[-1].unit->n_bonds-1]->tail.atom_num+
                main[-1].unit->list_num].position;
       b1a=atom[list_num + main->unit->head.atom_num].position;
       b1a.x = p1a.x - b1a.x;
       b1a.y = p1a.y - b1a.y;
       b1a.z = p1a.z - b1a.z;
     }
 /* add on unit */
     grow_backwards = TRUE;
     do_unit_sub(&list_num, n0, n1, n2, logrosen, main->unit, t, 1,
atom, twig,
                p1, b1, p0, b0, e, p, b, new);
     grow_backwards = FALSE;
 /* restore backbone unit */
     main->unit->n_atoms = n_atoms;
```

```
    main->unit->atom = q;
    for (j=0; j<n_bonds; j++) {
      main->unit->bond[j]->tail.axis = b2[j];
/* change head vector */
    if (!new && i > 0)
      b0 = get_main_b0(atom, main-1, 1);
    else if (new && i > 0)
      b0 = vector_scale(b[n_bonds-1], 1.0);
    p1 = p1a;
    b1 = b1a;
    }
  }
}
/* This routine creates the random positions.
   For new units, it picks and copies the winner.
*/
void do_unit_sub(int *list_num, int n0, int n1, int n2, double
*logrosen,
                 rigid_unit *unit, torsion_list *t, hbond_list *l,
                 atom_list *atom, vector *twig[], vector p1, vector
b1,
                 vector  p0,  vector  b0,  double  *e,  vector
p[MAX_BONDS],
                 vector b[MAX_BONDS], logical new)
{
  int i,j,i0;
  vector bond[KMAX][MAX_BONDS], point[KMAX][MAX_BONDS];
  double ftmp, cos_theta2, sin_theta2;
  double de[KMAX], sum, max;
  i0 = 0;
  if (!new) {
/* copy old configuration to first "guess" */
    i0 = 1;
    for (j=0; j<unit->n_atoms; j++)
      twig[0][j] = atom[*list_num + j].position;
  }
/* create gues s for new unit position */
  for (i=i0; i<KMAX; i++) {
    do {
        cos_theta2 = 1-2*ran2(1.0);
        sin_theta2 = 1-2*ran2(1.0);
        ftmp = cos_theta2*cos_theta2 + sin_theta2*sin_theta2;
      } while (ftmp > 1.0);
      ftmp = sqrt(ftmp);
      cos_theta2 /= ftmp;
      sin_theta2 /= ftmp;
      add_rigid_unit(unit, twig[i], p1, b1,
                     p0, b0, point[i], bond[i],
                     cos_theta2, sin_theta2);
    }
/* calculate probabilties -- be careful about zero of energy &
overflows */
  max = -1E99;
  for (j=0; j<KMAX; j++) {
    de[j] = -BETA * delta_energy(t, 1, atom, twig[j], *list_num,
n0, n1, n2,
                                  unit->n_atoms);
    if (de[j] > max) max = de[j];
  }
  sum = 0.0;
  for (j=0; j<KMAX; j++) {
    de[j] = exp(de[j] - max);
    sum += de[j];
  }
  *logrosen += log(sum) + max - log(KMAX);
  if (!new) {
/* determine points */
    for (j=0; j<unit->n_bonds; j++) {
              p [ j ]   =   a t o m [ * l i s t _ n u m   +
unit->bond[j]->tail.atom_num].position;
      b[j] = atom[unit->bond[j]->next->list_num +
                unit->bond[j]->next->head.atom_num].position;
      b[j].x -= p[j].x;
      b[j].y -= p[j].y;
      b[j].z -= p[j].z;
      b[j] = vector_scale(b[j], 1.0);
    }
    *list_num += unit->n_atoms;
  } else {
/* pick winner */
    de[0] /= sum;
    for (j=1; j<KMAX; j++) de[j] = de[j-1] + de[j]/sum;
    ftmp = ran2(1.0);
    for (i=0; i<KMAX; i++) if (ftmp <= de[i]) break;
    ftmp = de[i];
    if (i > 0) ftmp -= de[i-1];
    ftmp *= sum;
    *e -= (log(ftmp)+max)/BETA;
/* copy winner to atom array */
    for (j=0; j<unit->n_atoms; j++, (*list_num)++)
       atom[*list_num].position = twig[i][j];
    for (j=0; j<unit->n_bonds; j++) {
      p[j] = point[i][j];
      b[j] = bond[i][j];
    }
  }
}
/* This routine adds a rigid unit to the peptide structure
*/
void add_rigid_unit(rigid_unit *unit, vector *pos,
                    vector p1, vector b1, vector p0,
                    vector b0, vector point[MAX_BONDS],
                    vector bond[MAX_BONDS],
                    double cos_theta2, double sin_theta2)
{
  int i;
  double bond_len, cos_theta, sin_theta;
  vector n, r0;
  bond_len = vector_length(b1);
  r0.x = p0.x + b0.x*bond_len;
  r0.y = p0.y + b0.y*bond_len;
  r0.z = p0.z + b0.z*bond_len;
  b1.x /= bond_len;
  b1.y /= bond_len;
  b1.z /= bond_len;
  n = vector_cross(b1,b0);
  cos_theta = vector_dot(b0,b1);
  sin_theta = vector_length(n);
  if (sin_theta < EPS) {
    n.x = 1.0;
  } else {
    n.x /= sin_theta;
    n.y /= sin_theta;
    n.z /= sin_theta;
  }
  for (i=0; i<unit->n_atoms; i++)
    pos[i] = align(unit->atom[i].position, r0, p1,
                   n, cos_theta, sin_theta,
                   b0, cos_theta2, sin_theta2);
  for (i=0; i<unit->n_bonds; i++)
    point[i] = pos[unit->bond[i]->tail.atom_num];
  r0.x = 0.0; r0.y = 0.0; r0.z = 0.0; p1=r0;
  for (i=0; i<unit->n_bonds; i++)
    bond[i] = align(unit->bond[i]->tail.axis, r0, p1,
                    n, cos_theta, sin_theta,
                    b0, cos_theta2, sin_theta2);
}
/* This routine aligns the position
*/
vector align(vector p, vector r0, vector r1, vector n, double
cos_theta,
             double sin_theta, vector n2, double cos_theta2,
             double sin_theta2)
{
  vector ret;
  ret.x = p.x - r1.x;
  ret.y = p.y - r1.y;
  ret.z = p.z - r1.z;
  ret = vector_rotate(ret, n, cos_theta, sin_theta);
  ret = vector_rotate(ret, n2, cos_theta2, sin_theta2);
  ret.x += r0.x;
  ret.y += r0.y;
  ret.z += r0.z;
  return(r t);
}
```

```
*******************************************************************
                ENERGY DETERMINATION - PEPTIDE4.C
*******************************************************************

/*                          The energy routines
*/
#include "peptide.h"
#define N0 8
#define N1 11
#define N2 81
#define N3 84
#define N2 63
#define N3 66
#define SCALE 100
/* This energy routine tries to force a S-S ring-closure for
CAAAAAAC
*/
double zenergy(torsion_list *t, hbond_list *l, atom_list *atom,
              int n_atoms_total)
{
  double r1, r2;
  vector x, y, v;
  x = atom[N1].position;
  x.x -= atom[N0].position.x;
  x.y -= atom[N0].position.y;
  x.z -= atom[N0].position.z;
  x = vector_scale(x, 2.038);
  x.x += atom[N0].position.x;
  x.y += atom[N0].position.y;
  x.z += atom[N0].position.z;
  y = atom[N3].position;
  y.x -= atom[N2].position.x;
  y.y -= atom[N2].position.y;
  y.z -= atom[N2].position.z;
  y = vector_scale(y, 2.038);
  y.x += atom[N2].position.x;
  y.y += atom[N2].position.y;
  y.z += atom[N2].position.z;
  v = x;
```

```
    v.x -= atom[N2].position.x;
    v.y -= atom[N2].position.y;
    v.z -= atom[N2].position.z;
    r1 = vector_length2(v);
    v = y;
    v.x -= atom[N0].position.x;
    v.y -= atom[N0].position.y;
    v.z -= atom[N0].position.z;
    r2 = vector_length2(v);
    return(SCALE*(r1+r2)/BETA);
}
/* This energy routine tries to force a S-S ring-closure for
CAAAAAAC
*/
double zdelta_energy(torsion_list *t, hbond_list *l, atom_list
*atom,
                    vector *twig, int n_atoms, int n0, int n1, int
n2,
                    int n_twig)
{
  double r1, r2;
  vector x, y, v;
  r1 = r2 = 0.0;
  if (INTERVAL(N0, n_atoms, n_atoms+n_twig) &&
      INTERVAL(N2, n1, n2)) {
    x = twig[N1-n_atoms];
    x.x -= twig[N0-n_atoms].x;
    x.y -= twig[N0-n_atoms].y;
    x.z -= twig[N0-n_atoms].z;
    x = vector_scale(x, 2.038);
    x.x += twig[N0-n_atoms].x;
    x.y += twig[N0-n_atoms].y;
    x.z += twig[N0-n_atoms].z;
    y = atom[N3].position;
    y.x -= atom[N2].position.x;
    y.y -= atom[N2].position.y;
    y.z -= atom[N2].position.z;
    y = vector_scale(y, 2.038);
    y.x += atom[N2].position.x;
```

```
      y.y += atom[N2].position.y;
      y.z += atom[N2].position.z;
      v = x;
      v.x -= atom[N2].position.x;
      v.y -= atom[N2].position.y;
      v.z -= atom[N2].position.z;
      r1 = vector_length2(v);
      v = y;
      v.x -= twig[N0-n_atoms].x;
      v.y -= twig[N0-n_atoms].y;
      v.z -= twig[N0-n_atoms].z;
      r2 = vector_length2(v);
    } else if (INTERVAL(N2, n_atoms, n_atoms+n_twig) &&
               INTERVAL(N0, n0, n_atoms)) {
      x = atom[N1].position;
      x.x -= atom[N0].position.x;
      x.y -= atom[N0].position.y;
      x.z -= atom[N0].position.z;
      x = vector_scale(x, 2.038);
      x.x += atom[N0].position.x;
      x.y += atom[N0].position.y;
      x.z += atom[N0].position.z;
      y = twig[N3-n_atoms];
      y.x -= twig[N2-n_atoms].x;
      y.y -= twig[N2-n_atoms].y;
      y.z -= twig[N2-n_atoms].z;
      y = vector_scale(y, 2.038);
      y.x += twig[N2-n_atoms].x;
      y.y += twig[N2-n_atoms].y;
      y.z += twig[N2-n_atoms].z;
      v = x;
      v.x -= twig[N2-n_atoms].x;
      v.y -= twig[N2-n_atoms].y;
      v.z -= twig[N2-n_atoms].z;
      r1 = vector_length2(v);
      v = y;
      v.x -= atom[N0].position.x;
      v.y -= atom[N0].position.y;
      v.z -= atom[N0].position.z;
    r2 = vector_length2(v);
   }
   return(SCALE*(r1+r2)/BETA);
 }
 /* This routine returns the Coulomb, LJ, H-bond, and torsion
energies
    between the atoms in *atom and the atoms in *twig.
    The atoms in *twig must be those directly following those in
*atom.
    The atoms n_atoms to n_atoms+n_twig are in twig.
    The atoms n0 to n_atoms and n1 to n2 are in atom.
    n0 <= n_atoms <= n1 <= n2
 */
double delta_energy(torsion_list *t, hbond_list *l, atom_list
*atom,
                   vector *twig, int n_atoms, int n0, int n1, int
n2,
                   int n_twig)
{
  return(
         d_nonbond_energy(t, atom, twig, n_atoms, n0, n1, n2,
n_twig) +
         d_hbond_energy(l, atom, twig, n_atoms, n0, n1, n2, n_twig)
+
         d_torsion_energy(t, atom, twig, n_atoms, n0, n1, n2,
n_twig)
        );
}
/* This routine returns the total energy
*/
double energy(torsion_list *t, hbond_list *l, atom_list *atom,
             int n_atoms_total)
{
  return(
         nonbond_energy(t, atom, n_atoms_total) +
         hbond_energy(l, atom) +
         torsion_energy(t, atom)
        );
}
```

```
/* This routine returns the Coulomb and LJ energies
   between the atoms in *atom and the atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
*/
double d_nonbond_energy(torsion_list *t, atom_list *atom, vector
*twig,
                        int n_atoms, int n0, int n1, int n2, int
n_twig)
{
#define FACT 332.06 /* converts from ei ej/rij to Kcal/mol */
  int i, j, k;
  vector r;
  double r2, r6, e, eij, rij, rij3, term, a, b;
  e = 0.0;
  for (i=n0; i<n2; i++) {
    if (INTERVAL(i,n_atoms,n1)) continue;
    for (j=0; j<n_twig; j++) {
      r.x = atom[i].position.x - twig[j].x;
      r.y = atom[i].position.y - twig[j].y;
      r.z = atom[i].position.z - twig[j].z;
      r2 = vector_length2(r);
      r6 = r2*r2*r2;
      eij = sqrt(atom[i].p->ei * atom[n_atoms+j].p->ei);
      rij = 0.5*(atom[i].p->ri + atom[n_atoms+j].p->ri);
      rij3 = rij*rij*rij;
      a = eij * rij3*rij3*rij3*rij3;
      b = 2*eij * rij3*rij3;
/* epsilon = 4*r */
      term = FACT * atom[i].p->charge * atom[n_atoms+j].p->charge
/ (4*r2)
             +   a/(r6*r6)  - b/r6;
      e += term;
    }
  }
/* subtract off 1/2 of 1-4 interactions */
  for (; t; t=t->next)
  {
    i = t->num[0]; j = t->num[3];
```

```
      if (INTERVAL(i,n_atoms,n_atoms+n_twig)) {
        k = i;
        i = j;
        j = k;
      }
       if   (INTERVAL(j,n_atoms,n_atoms+n_twig)    &&
 (INTERVAL(i,n0,n_atoms) ||
          INTERVAL(i, n1, n2))) {
       r.x = atom[i].position.x - twig[j-n_atoms].x;
       r.y = atom[i].position.y - twig[j-n_atoms].y;
       r.z = atom[i].position.z - twig[j-n_atoms].z;
       r2 = vector_length2(r);
       r6 = r2*r2*r2;
       eij = sqrt(atom[i].p->ei * atom[j].p->ei);
       rij = 0.5 * (atom[i].p->ri + atom[j].p->ri);
       rij3 = rij*rij*rij;
       a = eij * rij3*rij3*rij3*rij3;
       b = 2*eij * rij3*rij3;
       term = FACT * atom[i].p->charge * atom[j].p->charge / (4*r2)
              + a/(r6*r6) - b/r6;
       e -= 0.5 * term;
     }
   }
   return(e);
#undef FACT
}
/* This routine returns the Coulomb and LJ energies
*/
double  nonbond_energy(torsion_list  *t,  atom_list  *atom,  int
n_atoms_total)
{
#define FACT 332.06 /* converts from ei ej/rij to Kcal/mol */
  int i, j;
  vector r;
  double r2, r6, e, eij, rij, rij3, term, a, b;
  e = 0.0;
  for (i=0; i<n_atoms_total; i++)
    for (j=i+1; j<n_atoms_total; j++) {
      r.x = atom[i].position.x - atom[j].position.x;
```

```
        r.y = atom[i].position.y - atom[j].position.y;
        r.z = atom[i].position.z - atom[j].position.z;
        r2 = vector_length2(r);
        r6 = r2*r2*r2;
        eij = sqrt(atom[i].p->ei * atom[j].p->ei);
        rij = 0.5*(atom[i].p->ri + atom[j].p->ri);
        rij3 = rij*rij*rij;
        a = eij * rij3*rij3*rij3*rij3;
        b = 2*eij * rij3*rij3;
/* epsilon = 4*r */
        term = FACT * atom[i].p->charge * atom[j].p->charge / (4*r2)
               +    a/(r6*r6)  - b/r6;
        e += term;
      }
/* subtract off 1/2 of 1-4 interactions */
  for (; t; t=t->next)
  {
    i = t->num[0]; j = t->num[3];
    r.x = atom[i].position.x - atom[j].position.x;
    r.y = atom[i].position.y - atom[j].position.y;
    r.z = atom[i].position.z - atom[j].position.z;
    r2 = vector_length2(r);
    r6 = r2*r2*r2;
    eij = sqrt(atom[i].p->ei * atom[j].p->ei);
    rij = 0.5 * (atom[i].p->ri + atom[j].p->ri);
    rij3 = rij*rij*rij;
    a = eij * rij3*rij3*rij3*rij3;
    b = 2*eij * rij3*rij3;
    term = FACT * atom[i].p->charge * atom[j].p->charge / (4*r2)
           + a/(r6*r6) - b/r6;
    e -= 0.5 * term;
  }
  return(e);
#undef fact
}
/* This routine returns the H-bond energy
   betw en the atoms in *atom and th  atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
```

```
 */
 double d_hbond_energy(hbond_list *l, atom_list *atom, vector *twig,
                         int n_atoms, int n0, int n1, int n2, int
 n_twig)
 {
   int i,j,k;
   vector r;
   double r2, e;
   e = 0.0;
   for (; l; l=l->next) {
     i = l->num[0]; j = l->num[1];
     if (INTERVAL(i,n_atoms,n_atoms+n_twig)) {
       k = i;
       i = j;
       j = k;
     }
      if    (INTERVAL(j,n_atoms,n_atoms+n_twig)     &&
(INTERVAL(i,n0,n_atoms) ||
         INTERVAL(i,n1,n2))) {
       r.x = atom[i].position.x - twig[j-n_atoms].x;
       r.y = atom[i].position.y - twig[j-n_atoms].y;
       r.z = atom[i].position.z - twig[j-n_atoms].z;
       r2 = vector_length2(r);
        e    +=    l->p->a    /    (r2*r2*r2*r2*r2*r2)
l->p->b/(r2*r2*r2*r2*r2);
     }
   }
   return(e);
}
/* This routine returns the H-bond energy
*/
double hbond_energy(hbond_list *l, atom_list *atom)
{
  vector r;
  double r2, e;
  e = 0.0;
  for (; l; l=l->next) {
    r.x = atom[l->num[0]].position.x - atom[l->num[1]].position.x;
    r.y = atom[l->num[0]].position.y - atom[l->num[1]].position.y;
    r.z = atom[l->num[0]].position.z - atom[l->num[1]].position.z;
    r2 = vector_length2(r);
    e += l->p->a / (r2*r2*r2*r2*r2*r2) - l->p->b/(r2*r2*r2*r2*r2);
  }
  return(e);
}
/* This routine returns the H-bond energy
   between the atoms in *atom and the atoms in *twig.
   The atoms in *twig must be those directly following those in
*atom.
*/
double d_torsion_energy(torsion_list *t, atom_list *atom, vector
*twig,
                        int n_atoms, int n0, int n1, int n2, int
n_twig)
{
  int i,j,k,l;
  vector v[4];
  double theta, e, tmp;
  e = 0.0;
  for (; t; t=t->next)
  {
    if (t->p->v0[0] != 0.0 || t->p->v0[1] != 0.0 || t->p->v0[2] !=
0.0) {
      i = t->num[0]; j = t->num[1]; k = t->num[2]; l = t->num[3];
      if (INTERVAL(i,n_atoms+n_twig,n1) || i >= n2 || i < n0)
continue;
      if (INTERVAL(j,n_atoms+n_twig,n1) || j >= n2 || j < n0)
continue;
      if (INTERVAL(k,n_atoms+n_twig,n1) || k >= n2 || k < n0)
continue;
      if (INTERVAL(l,n_atoms+n_twig,n1) || l >= n2 || l < n0)
continue;
      if (!(INTERVAL(i,n_atoms,n_atoms+n_twig) ||
            INTERVAL(j,n_atoms,n_atoms+n_twig) ||
            INTERVAL(k,n_atoms,n_atoms+n_twig) ||
            INTERVAL(l,n_atoms,n_atoms+n_twig))) continue;
/*      printf("%d %d %d %d", i, j, k, l); */
      if (INTERVAL(i,n_atoms,n_atoms+n_twig))
```

```
        v[0] = twig[i-n_atoms]; else v[0] = atom[i].position;
      if (INTERVAL(j,n_atoms,n_atoms+n_twig))
        v[1] = twig[j-n_atoms]; else v[1] = atom[j].position;
      if (INTERVAL(k,n_atoms,n_atoms+n_twig))
        v[2] = twig[k-n_atoms]; else v[2] = atom[k].position;
      if (INTERVAL(l,n_atoms,n_atoms+n_twig))
        v[3] = twig[l-n_atoms]; else v[3] = atom[l].position;
      theta = torsion(v[0], v[1], v[2], v[3]);
      tmp = (t->p->v0[0]*(1 + cos(  theta-t->p->phi0[0])) +
             t->p->v0[1]*(1 + cos(2*theta-t->p->phi0[1])) +
             t->p->v0[2]*(1   +   cos(3*theta-t->p->phi0[2])))   /
t->degen;
/*      printf(" %lf %lf\n",theta,tmp); */
      e += tmp;
    }
  }
  return(e);
}
/* This routine returns the torsional energy
*/
double torsion_energy(torsion_list *t, atom_list *atom)
{
  double theta, e, tmp;
  e = 0.0;
  for (; t; t=t->next)
  {
    if (t->p->v0[0] != 0.0 || t->p->v0[1] != 0.0 || t->p->v0[2] !=
0.0) {
      theta    =    torsion(atom[t->num[0]].position,
atom[t->num[1]].position,
                           atom[t->num[2]].position,
atom[t->num[3]].position);
      tmp = (t->p->v0[0]*(1 + cos(  theta-t->p->phi0[0])) +
             t->p->v0[1]*(1 + cos(2*theta-t->p->phi0[1])) +
             t->p->v0[2]*(1   +   cos(3*theta-t->p->phi0[2])))   /
t->degen;
/*        printf("%d %d %d %d %lf %lf\n", t->num[0], t->num[1],
t->num[2],
                  t->num[3], theta, tmp); */
        e += tmp;
      }
    }
   return(e);
  }


******************************************************************
                  MONTE CARLO ROUTINES - PEPTIDES.C
******************************************************************

/*                          The Monte Carlo routines
*/
#include "peptide.h"
/* This routine drives the configurational bias Monte Carlo
*/
void do_mc(rigid_unit *unit, torsion_list *t, hbond_list *l,
             atom_list *atom, atom_list *atom2, atom_info *atom_tmp,
             vector *twig[], regrowth *main, regrowth *side,
             int n_amino_acids, int n_atoms_total, int n_main, int
n_side,
             logical cyclic)
{
  int list_num, i, j;
  double logrosen, e, e2, emin;
  vector p0, b0;
  vector v1,v2;
  emin = 1.0E99;
  list_num = 0;
  p0.x = 0.0; p0.y = 0.0; p0.z = 0.0;
  b0.x = 0.0; b0.y = 0.0; b0.z = 1.0;
  e = 0;
  logrosen = 0;
/* create initial geomeotry */
  do_unit(&list_num, 0, n_atoms_total, n_atoms_total,
           &logrosen, unit, unit, t, l, atom, twig,
           p0, b0, &e);
/* read in initial geometry */
  if (0) read_restart(atom, n_atoms_total);
  if (cyclic)
```

```
    read_cycle(t, 1, atom, main, side, twig, n_main, n_side,
n_atoms_total);
/*
  do_backbone_f(0, n_main, n_atoms_total, &logrosen, main,
                side, t, 1, atom, twig, &e, TRUE);
  do_backbone_b(n_main-1, n_main, n_atoms_total, &logrosen, main,
                side, t, 1, atom, twig, &e, TRUE);
  do_backbone_f_rigid(0, n_main, n_atoms_total,
                      &logrosen, main,
                      side, t, 1, atom, atom_tmp, twig, &e, TRUE);
  do_backbone_b_rigid(n_main-1, n_main, n_atoms_total,
                      &logrosen, main,
                      side, t, 1, atom, atom_tmp, twig, &e, TRUE);
*/
  emin = e = energy(t, 1, atom, n_atoms_total);
/* copy old positions into new */
  for (j=0; j<n_atoms_total; j++) atom2[j] = atom[j];
/* do Monte Carlo */
  for (i=0; i<16000; i++) {
    printf("%d\n",i);
    rotate_main(atom, atom2, twig, main, side, t, 1, n_main,
    n_atoms_total, &e);
/*
    regrow_main(t, 1, atom, atom2, atom_tmp, twig, main, side,
                n_main, n_atoms_total, &e);
    regrow_side(t, 1, atom, atom2, twig, main, side,
                n_side, n_atoms_total, &e);
*/
    if (e < emin) {
      emin = e;
      write_car_file(n_amino_acids,   n_atoms_total,   atom,
"min.car");
    }
  }
  printf("emin %lf\n",emin);
}
/* This routine reads in a restart file
*/
void read_restart(atom_list *atom, int n_atoms_total)
```

```
{
#define LINELEN 200
   FILE *fp;
   int i;
   char name[30], line[LINELEN];
   strcpy(name, "restart.car");
   if ((fp = fopen(name, "r")) == NULL) {
     printf("Data file %s does not exist\n", name);
     exit(1);
   }
   fgets(line, LINELEN, fp);
   fgets(line, LINELEN, fp);
   fgets(line, LINELEN, fp);
   fgets(line, LINELEN, fp);
   for (i=0; i<n_atoms_total; i++) {
     fgets(line, LINELEN, fp);
     sscanf(line, "%s %lf %lf %lf", name,
                                        &atom[i].position.x,
                                        &atom[i].position.y,
                                        &atom[i].position.z);
   }
   fclose(fp);
}
/* This routine reads in the backbone units plus one side-chain
atom
   for the geometry CXXXXXXC.  It then adds on each of the side
   groups randomly
*/
void read_cycle(torsion_list *t, hbond_list *l,
                atom_list *atom, regrowth *main, regrowth *side,
                vector *twig[], int n_main, int n_side, int
n_atoms_total)
{
#define LINELEN 200
  FILE *fp;
  int i, j, k, list_num;
  char name[30], line[LINELEN];
  double logrosen,  ;
/* read in loop atoms plus one side group atom */
```

```
  if (n_main != 2*8+3) {
    printf("This cyclic geometry is not supported\n");
    exit(1);
  }
  strcpy(name, "CX6C.car");
  if ((fp = fopen(name, "r")) == NULL) {
    printf("Data file %s does not exist\n", name);
    exit(1);
  }
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  fgets(line, LINELEN, fp);
  for (i=0; i<n_main; i++) {
    /* printf("%d\n",main[i].unit->list_num); */
    for (j=0; j<main[i].unit->n_atoms; j++) {
      k = main[i].unit->list_num + j;
      fgets(line, LINELEN, fp);
      sscanf(line, "%s %lf %lf %lf", name,
                                    &atom[k].position.x,
                                    &atom[k].position.y,
                                    &atom[k].position.z);
      /* printf("%d %s %lf %lf %lf\n",k,name,
                                    atom[k].position.x,
                                    atom[k].position.y,
                                    atom[k].position.z); */
    }
    if (main[i].unit->n_bonds == 2) {
      k++;
      fgets(line, LINELEN, fp);
      sscanf(line, "%s %lf %lf %lf", name, &atom[k].position.x,
                                    &atom[k].position.y,
                                    &atom[k].position.z);
      /* printf("%d %s %lf %lf %lf\n",k,name,
                                    atom[k].position.x,
                                    atom[k].position.y,
                                    atom[k].position.z); */
    }
  }
```

```
   fclose(fp);
 /* add on side groups */
   for (i=0; i<n_side; i++) {
     list_num = side[i].unit->list_num;
     do_unit(&list_num, 0, n_atoms_total, n_atoms_total,
           &logrosen, side[i].unit, side[i].unit, t, l, atom, twig,
           get_side_p0(atom, side, i), get_side_b0(atom, side, i),
           &e);
   }
 }
 /* This routine regrows from a main chain unit onwards
 */
 void regrow_main(torsion_list *t, hbond_list *l,
                  atom_list *atom, atom_list *atom2,
                  atom_info *atom_tmp, vector *twig[],
                  regrowth *main, regrowth *side,
                  int n_main, int n_atoms_total, double *e)
 {
   logical forward;
   int list_num, i, j, k;
   double logrosen1, logrosen2, x, e2, e1;
 /* pick main group to start regrowth from */
   i = n_main*ran2(1.0);
 /* pick direction to regrow */
   forward = (ran2(1.0) > 0.5);
   printf("regrowing %s from unit %d\n",(forward) ? "forward" :
 "backward", i);
   list_num = main[i].unit->list_num;
 /* copy old positions into new */
   for   (j=0;   j<n_atoms_total;   j++)   atom2[j].position   =
 atom[j].position;
 /* regrow new peptide */
   e2 = 0;
   logrosen2 = 0.0;
   if (forward)
     do_backbone_f_rigid(i, n_main, n_atoms_total, &logrosen2, main,
                         side, t, l, atom2, atom_tmp, twig, &e2,
 TRUE);
   else
```

```
    do_backbone_b_rigid(i, n_main, n_atoms_total, &logrosen2, main,
                              side, t, l, atom2, atom_tmp, twig, &e2,
TRUE);
  e2 = energy(t, l, atom2, n_atoms_total);
/* get old Rosenbluth weight */
  list_num = main[i].unit->list_num;
  e1 = 0.0;
  logrosen1 = 0.0;
  if (forward)
    do_backbone_f_rigid(i, n_main, n_atoms_total, &logrosen1, main,
                              side, t, l, atom, atom_tmp, twig, &e1,
FALSE);
  else
    do_backbone_b_rigid(i, n_main, n_atoms_total, &logrosen1, main,
                              side, t, l, atom, atom_tmp, twig, &e1,
FALSE);
  printf("Wn Wo %lf %lf\n",logrosen2, logrosen1);
  printf("En Eo %lf %lf\n",e2, *e);
/* perform acceptance test */
  x = 1.0;
  if (logrosen1 > logrosen2) x = exp(logrosen2-logrosen1);
/* accept new configuration */
  if (ran2(1.0) < x) {
    for   (j=0;   j<n_atoms_total;   j++)   atom[j].position   =
atom2[j].position;
    *e = e2;
    printf("SWAP\n");
  }
}
/* This routine regrows a side chain
*/
void regrow_side(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_side, int n_atoms_total, double *e)
{
  int list_num, i, j, k, n1;
  double logrosen1, logrosen2, x, e2;
  if (n_side ==0 ) return;
```

```
/* pick main group to start regrowth from */
  i = n_side*ran2(1.0);
  printf("regrowing side chain %d\n",i);
  list_num = side[i].unit->list_num;
  logrosen2 = 0.0;
/* copy old positions into new */
  for   (j=0;   j<n_atoms_total;   j++)   atom2[j].position   =
atom[j].position;
/* regrow side chain */
  e2 = 0;
/* determine n1 */
                             n        1                             =
side[i].prev->bond[side[i].prev->n_bonds-1]->next->list_num;
  do_unit(&list_num, 0, n1, n_atoms_total,
          &logrosen2, side[i].unit, side[i].unit, t, 1, atom2,
twig,
          get_side_p0(atom, side, i), get_side_b0(atom, side, i),
          &e2);
  e2 = energy(t, 1, atom2, n_atoms_total);
/* get old Rosenbluth weight */
  list_num = side[i].unit->list_num;
  logrosen1 = 0.0;
  old_unit(&list_num, 0, n1, n_atoms_total, &logrosen1,
           side[i].unit, side[i].unit, t, 1, atom, twig,
           get_side_p0(atom, side, i), get_side_b0(atom, side, i));
  printf("Wn Wo %lf %lf\n",logrosen2, logrosen1);
  printf("En Eo %lf %lf\n",e2, *e);
/* perform acceptance test */
  x = 1.0;
  if (logrosen1 > logrosen2) x = exp(logrosen2-logrosen1);
/* accept new configuration */
  if (ran2(1.0) < x) {
    for (j=side[i].unit->list_num; j<list_num; j++)
      atom[j].position = atom2[j].position;
    *e = e2;
    printf("SWAP\n");
  }
}
```

```
*****************************************************************
              CONCERTED ROTATION ROUTINES - PEPTIDE6.C
*****************************************************************

/*                      The concerted rotation routines
*/
#include "peptide.h"
/* global variables */
vector l[8], r[8];
double theta[8], m[3][3];
logical head[8];
/* This routine performs a concerted rotation on part of the main
chain.
*/
void rotate_main(atom_list *atom, atom_list *atom2, vector
                 *twig[], regrowth *main, regrowth *side,
                 torsion_list *t, hbond_list *l, int n_main, int
                 n_atoms_total, double *e)
{
  double jo, jn, logroseno, logrosenn, x, phi1, eo, en;
  int no, nn, i, j, i1, i2, i0;
  vector q;

  logical valid[4];
  double phi2[4], phi3[4], phi4[4], f[4];

  i0 = n_main * ran2(1.0);
  printf("Rotating from position %d\n",i0);
/* copy atom positions to atom2 */
  for   (i=0;   i<n_atoms_total;   i++)   atom2[i].position   =
atom[i].position;
/* determine theta, r, l */
  get_rot_params(atom, main, i0, n_main);
/* get original jacobian */
  jo = jac(atom, main, i0, n_main);
/* get constants needed by F5 */
  F5init(get_main_b0(atom, main, (i0+1) % n_main), &phi1);
/* get original Rosenbluth weight */
  eo = energy(t, l, atom, n_atoms_total);
```

```
get_rot_rosenbluth(atom, atom2, twig, main, t, 1, i0, n_main,
                   n_atoms_total, &no, &j, &logroseno, &en);

printf("%d\n",no);
if (no == 0) return; /* should never happen */
/* rotate r1 and get new constants */
q = rotate_r1(atom, main, i0, n_main);
F5init(q, &phi1);
/* get new Rosenbluth weight */
get_rot_rosenbluth(atom, atom2, twig, main, t, 1, i0, n_main,
                   n_atoms_total, &nn, &j, &logrosenn, &en);

printf("%d\n",nn);
if (nn == 0) return; /* geometric failure */
/* copy atomic positions */
i1 = main[i0].unit->list_num;
i2 = main[(i0+7) % n_main].unit->list_num;
if (i2 < i1) i2 += n_atoms_total;
for (i=i1; i<i2; i++)
  atom2[i % n_atoms_total].position = twig[j][i % n_atoms_total];
/* determine new Jacobian */
jn = jac(atom2, main, i0, n_main);
/* Doros move */
/* x = exp(-BETA*(en-eo)) * jn/jo * nn/no; */
/* CBMC move */
if (logrosenn - logroseno < -10.0)
  x = 0.0;
else if (logrosenn - logroseno > 10.0)
  x = 1.0;
else
  x = jn/jo * exp(logrosenn - logroseno);
/* decide if move is accepted */
printf("Wn Wo %lf %lf\n",logrosenn, logroseno);
printf("En Eo %lf %lf\n",en, eo);
if (ran2(1.0) < x) {
  printf("SWAP\n");
  *e = en;
  /* copy atomic positions */
  i1 = main[i0].unit->list_num;
  i2 = main[(i0+7) % n_main].unit->list_num;
  if (i2 < i1) i2 += n_atoms_total;
```

```
    for (i=i1; i<i2; i++)
      atom[i   %   n_atoms_total].position   =   twig[j][i   %
n_atoms_total];
    } els
    *e = eo;
}
/* This routine gets the theta, r, and l parameters */
void get_rot_params(atom_list *atom, regrowth *main, int i0,
                    int n_main)
{
  int i;
  vector t, v, v2;
  double len;
  rigid_unit *unit, *unit2, *unit3;
/* determine theta */
  for (i=0; i<8; i++) {
    unit = main[(i+i0) % n_main].unit;
    theta[i] = vector_dot(unit->head.axis,
                          unit->bond[unit->n_bonds-1]->tail.axis)
/
                          vector_length(unit->head.axis);
    theta[i] = (theta[i] < 1.0-EPS) ? acos(theta[i]) : 0.0;    }
/* determine r */
  for (i=0; i<8; i++) head[i] = TRUE;
  if (fabs(theta[5]) < EPS) head[5] = FALSE;
  for (i=0; i<8; i++) {
    unit = main[(i+i0) % n_main].unit;
    r[i] = atom[unit->list_num + ((head[i]) ? unit->head.atom_num
:
          unit->bond[unit->n_bonds-1]->tail.atom_num)].position;
  }
/* determine l */
  for (i=1; i<8; i++) {
    t.x = r[i].x - r[i-1].x;
    t.y = r[i].y - r[i-1].y;
    t.z = r[i].z - r[i-1].z;
    len = vector_length(t);
    /* if (2.03<len && len <2.05) len = 2.038;
    t = vector_scale(t, len); */
    l[i].x = len; l[i].y = l[i].z = 0.0;
    if (((main[(i+i0) % n_main].prev->type == Cunit) &&
         head[i-1]) || !head[i]) {
      l[i].x = vector_dot(t, get_main_b0(atom, main, (i+i0) %
n_main));
      l[i].y = sqrt(len * len - l[i].x * l[i].x);
    }
  }
/*
  for (i=1; i<8; i++) printf("%d %lf %lf %lf %lf\n",i, theta[i],
                               l[i].x, l[i].y, l[i].z);
  for (i=1; i<8; i++)
    printf("%d %lf %lf %lf\n",i, r[i].x, r[i].y, r[i].z);
*/
}
/* This routine checks the rigid unit theta values
*/
void check_theta(atom_list *atom, regrowth *main, int n_main)
{
  int i;
  vector t, v, v2, r;
  double len, theta;
  rigid_unit *unit, *unit2, *unit3;
  for (i=0; i<n_main; i++) {
    unit = main[i % n_main].unit;
    unit2 = main[i % n_main].prev;
    unit3 = main[(i+1) % n_main].unit;
    r = atom[unit->list_num + unit->head.atom_num].position;
    t = atom[unit2->list_num +

unit2->bond[unit2->n_bonds-1]->tail.atom_num].position;
    t.x = r.x - t.x; t.y = r.y - t.y; t.z = r.z - t.z;
      printf("%lf   %lf   ",   vector_length(t),
vector_length(unit->head.axis));
    v = atom[unit3->list_num + unit3->head.atom_num].position;
    v2 = atom[unit->list_num +
         unit->bond[unit->n_bonds-1]->tail.atom_num].position;
    v.x -= v2.x; v.y -= v2.y; v.z -= v2.z;
    theta = v ctor_dot(t, v) / (vector_length(v)*v ctor_length(t));
```

```
      theta = (theta < 1.0-EPS) ? acos(theta) : 0.0;
      printf("%d %lf ",i, theta);
      theta = vector_dot(unit->head.axis,
                         unit->bond[unit->n_bonds-1]->tail.axis) /
                         vector_length(unit->head.axis);
      theta = (theta < 1.0-EPS) ? acos(theta) : 0.0;
      printf("%lf \n",theta);
    }
  }
  /* This routine determines the Rosenbluth weight */
  void get_rot_rosenbluth(atom_list *atom, atom_list *atom2,
                          vector *twig[], regrowth *main,
                          torsion_list *t, hbond_list *l, int i0,
                          int n_main, int n_atoms_total, int *n,
                          int *j, double *logrosen, double *e)
  {
    double phi[MAX_ROOTS][5], phi1, max, sum, de[MAX_ROOTS], ftmp;
    int i, k, k1, k2;
  /* get phi0-phi1 solutions */
    get_phi1(phi, n);
    if (*n == 0) return;
    if (*n > MAX_ROOTS) {
      printf("too many roots\n");
      *n = 0;
      return;
    }
  /* determine energies of solutions */
    max = -1E99;
    for (i=0; i<*n; i++) {
      get_r(phi[i][1], phi[i][2], phi[i][3], phi[i][4]);
      do_rotation(atom, twig[i], main, i0, n_main, n_atoms_total);
      k1 = main[i0].unit->list_num;
      k2 = main[(i0+7) % n_main].unit->list_num;
      if (k2 < k1) k2 += n_atoms_total;
      for (k=k1; k<k2; k++)
        atom2[k   %   n_atoms_total].position   =   twig[i][k   %
n_atoms_total];
      de[i] = -BETA*energy(t, l, atom2, n_atoms_total);
      if (de[i] > max) max = de[i];
    }
    sum = 0.0;
    for (i=0; i<*n; i++) {
      de[i] = exp(de[i] - max);
      sum += de[i];
    }
    *logrosen = log(sum) + max;
  /* pick winner */
  /* Doros move */
    /* *j = *n*ran2(1.0); */
  /* CBMC move */
    de[0] /= sum;
    for (i=1; i<*n; i++) de[i] = de[i-1] + de[i]/sum;
    ftmp = ran2(1.0);
    for (*j=0; *j<*n; (*j)++) if (ftmp <= de[*j]) break;
  /* get energy of winner */
    ftmp = de[*j];
    if (*j > 0) ftmp -= de[*j-1];
    ftmp *= sum;
    *e = -(log(ftmp)+max)/BETA;
  /* assign r to the winner */
    get_r(phi[*j][1], phi[*j][2], phi[*j][3], phi[*j][4]);
  }
  /* This routine calculates the jacobian
  */
  double jac(atom_list *atom, regrowth *main, int i0, int n_main)
  {
    int i;
    vector u[7], h[6], t, v;
    double b[5][5];

  /* form ui and hi */
    for  (i=1;  i<7;  i++)  u[i]  =  get_main_b0(atom,  main,  (i0+i)
  %n_main);
    for (i=1; i<5; i++) h[i] = r[i];
    h[5] = atom[main[(i0+5)%n_main].unit->list_num +
                main[(i0+5)%n_main].unit->head.atom_num].position;
    v.x = r[6].x - h[5].x; v.y = r[6].y - h[5].y;
    v.z = r[6].z - h[5].z;
    v = vector_scale(v, 1.0);
  /* form B matrix */
    for (i=1; i<6; i++) {
      t.x = r[5].x - h[i].x;
      t.y = r[5].y - h[i].y;
      t.z = r[5].z - h[i].z;
      t = vector_cross(u[i], t);
      b[0][i-1] = t.x;
      b[1][i-1] = t.y;
      b[2][i-1] = t.z;
    }
    for (i=1; i<6; i++) {
      t = vector_cross(u[i], u[6]);
      b[3][i-1] = t.x;
      b[4][i-1] = t.y;
    }
    return(1.0/fabs(det5(b)));
  }
  /* This routine rotates phi0 to change r[1].
     It returns the new b0 for unit i0+1.
  */
  vector rotate_r1(atom_list *atom, regrowth *main, int i0, int
                   n_main)
  {
    double c, s, y;
    vector x, n;
  /* choose delta phi0 */
    y = DPHI * (1-2*ran2(1.0));
    c = cos(y);
    s = sin(y);
    n = get_main_b0(atom, main, i0);
  /* rotate about axis */
    x = r[1];
    x.x -= r[0].x;
    x.y -= r[0].y;
    x.z -= r[0].z;
    x = vector_rotate(x, n, c, s);
    r[1].x = r[0].x + x.x;
    r[1].y = r[0].y + x.y;
    r[1].z = r[0].z + x.z;
  /* compute new b0 for unit i0+1 */
    return(vector_rotate(get_main_b0(atom, main, (i0+1) % n_main),
  n, c, s));
  }
  /* This routine constructs r2-r4 from the theta, phi
  information */
  void get_r(double phi1, double phi2, double phi3, double phi4)
  {
    int i;
    vector x, y;
  /*
    printf("\n");
    printf("%lf %lf %lf %lf %lf\n", phi1, phi2, phi3, phi4);
  */
    x = bxm(m, l[1]);
    r[1].x = x.x + r[0].x;
    r[1].y = x.y + r[0].y;
    r[1].z = x.z + r[0].z;
    x = bxm(m, flory_rot(theta[1], phi1, l[2]));
    r[2].x = x.x + r[1].x;
    r[2].y = x.y + r[1].y;
    r[2].z = x.z + r[1].z;
    x = bxm(m, flory_rot(theta[1], phi1, flory_rot(theta[2], phi2,
  l[3])));
    r[3].x = x.x + r[2].x;
    r[3].y = x.y + r[2].y;
    r[3].z = x.z + r[2].z;
    x = bxm(m, flory_rot(theta[1], phi1, flory_rot(theta[2],
            phi2, flory_rot(theta[3], phi3, l[4]))));
    r[4].x = x.x + r[3].x;
    r[4].y = x.y + r[3].y;
    r[4].z = x.z + r[3].z;
  /*
    for (i=1; i<7; i++)
       printf("%d %lf %lf %lf\n",i, r[i].x, r[i].y, r[i].z);
  */
  }
  /* This routine rotates the rigid units to the positions
```

```
   of the concerted rotation.
*/
void do_rotation(atom_list *atom, vector *twig, regrowth *main,
                 int i0, int n_main, int n_atoms_total)
{
  int i, j, i1, i2, i3, j2;
  double m[3][3], a[3][3], tmp, len2;
  vector x1, x2, y1, y2, x;
  rigid_unit *unit;
  for (i=-1; i<6; i++) {
  i1 = (i+i0+n_main) % n_main;
  i2 = (i+i0+1) % n_main;
  i3 = (i+i0+2) % n_main;
/* get x1 & x2 */
  x1 = r[i+1];
  x = (i > -1) ?

twig[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
             main[i1].unit->list_num] :

atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
             main[i1].unit->list_num].position;
    x1.x -=  x.x;  x1.y -= x.y; x1.z -= x.z;
    x2 = atom[main[i2].unit->list_num + ((head[i+1]) ?
              main[i2].unit->head.atom_num :

main[i2].unit->bond[main[i2].unit->n_bonds-1]->tail.atom_num)]
              .position;
                                           x                    =
atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num
+
              main[i1].unit->list_num].position;
    x2.x -= x.x; x2.y -= x.y; x2.z -= x.z;
/* get rotation matrix */
    flory_lab(a, x1, x2);
/* get y1 & y2 */
    y1 = r[i+2];
    x = (i > -1) ?
```

```
 twig[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
                main[i1].unit->list_num]  :

 atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
                main[i1].unit->list_num].position;
     y1.x -= x.x; y1.y -= x.y; y1.z -= x.z;
     y2 = atom[main[i3].unit->list_num + ((head[i+2]) ?
                 main[i3].unit->head.atom_num :

 main[i3].unit->bond[main[i3].unit->n_bonds-1]->tail.atom_num)]
                 .position;
                                            x                       =
 atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num
 +
               main[i1].unit->list_num].position;
     y2.x -= x.x; y2.y -= x.y; y2.z -= x.z;
     y2 = mxb(a, y2);
 /* get projection */
     len2 = vector_length2(x1);
     tmp = vector_dot(y2, x1) / len2;
     y2.x -= x1.x * tmp;
     y2.y -= x1.y * tmp;
     y2.z -= x1.z * tmp;
     tmp = vector_dot(y1, x1) / len2;
     y1.x -= x1.x * tmp;
     y1.y -= x1.y * tmp;
     y1.z -= x1.z * tmp;
 /* get rotation matrix */
     flory_lab(m, y1, y2);
     mxm (m, a);
 /* perform rotation */
                                       x      1                     =
 atom[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
               main[i1].unit->list_num].position;
     x2 = (i > -1) ?

 twig[main[i1].unit->bond[main[i1].unit->n_bonds-1]->tail.atom_num+
           main[i1].unit->list_num]  : x1;
     j2 = main[i3].unit->list_num;
```

```
      if (i3 == 0) j2 = n_atoms_total;
      for (j=main[i2].unit->list_num; j < j2; j++) {
        x = atom[j].position;
        x.x -= x1.x;
        x.y -= x1.y;
        x.z -= x1.z;
        x = mxb(m, x);
        x.x += x2.x;
        x.y += x2.y;
        x.z += x2.z;
        twig[j] = x;
      }
    }
  }
  /* This routine determines the phi1-phi3 values
  */
  void get_phi1(double phi[MAX_ROOTS][5], int *n)
  {
  #define NTRY 10000
    int i, j;
    logical valid[NTRY+1][4];
    double phi1[NTRY+1], phi2[4], phi3[4], phi4[4];
    double f[NTRY+1][4];
    *n = 0;
    i = 0;
  /* Evaluate F5 */
    for (i=0; i<=NTRY; i++) {
      phi1[i] = -PI + i*2*PI/NTRY;
      F5(phi1[i], phi2, phi3, phi4, f[i], valid[i]);
    }
  /* Now search for roots */
    for (i=0; i<NTRY; i++) {
      for (j=0; j<4; j++) {
        if (!valid[i][j] || !valid[i+1][j]) continue;
        if ((f[i][j] < 0 && f[i+1][j] > 0) ||
            (f[i][j] > 0 && f[i+1][j] < 0)) {
          if (*n >= MAX_ROOTS) {
            printf("Exc ssive   number   of   roots   failure   in
get_phi1\n");
```

```
            return;
          }
          get_root(phi1[i], phi1[i+1], &phi[*n][1], &phi[*n][2],
                   &phi[*n][3], &phi[*n][4], j);
          (*n)++;
        }
      }
    }
#undef NTRY
}
/* This routine refines a root using bisection
*/
void get_root(double x0, double x1, double *p1, double *p2,
double *p3,                  double *p4, int n)
{
  logical valid[4];
  double phi2[4], phi3[4], phi4[4], f[4];
/* order roots: f(x0) < 0 && f(x1) > 0 */
  F5(x1, phi2, phi3, phi4, f, valid);
  if (f[n] < 0.0) {
    *p1 = x0;
    x0 = x1;
    x1 = *p1;
  }
/* do bisection to refine root */
  do {
    *p1 = 0.5*(x1+x0);
    F5(*p1, phi2, phi3, phi4, f, valid);
    if (f[n] > 0) x1 = *p1; else x0 = *p1;
  } while (fabs(x1-x0) > EPS);
  *p2 = phi2[n];
  *p3 = phi3[n];
  *p4 = phi4[n];
}
/* constants */
double c10, c11, c12, q12, c20, c21, c22, fact1, fact2;
vector x0, u60;
/* This routine sets up constants that F5 uses.
   The constants are independent of phi1
```

```
*/
void F5init(vector q2, double *phi1)
{
  int i,j;
  vector t;
  double c1, c2, a[3][3], tmp;
  t.x = 1.0; t.y = t.z = 0.0;
  flory_labinv(m, q2, t);
  t.x = r[1].x - r[0].x; t.y = r[1].y - r[0].y; t.z = r[1].z -
r[0].z;
  t = mxb(m, t);
  if (fabs(t.y) < EPS && fabs(t.z) < EPS) {
    c1 = 1.0;
    c2 = 0.0;
  } else {
    c1 = (l[1].y*t.y + t.z*l[1].z)/(t.y*t.y + t.z*t.z);
    c2 = (-l[1].z*t.y + t.z*l[1].y)/(t.y*t.y + t.z*t.z);
    if (fabs(c1) < EPS && fabs(c2) < EPS) c1 = 1.0;
  }
  a[0][0] = 1; a[0][1] = 0; a[0][2] = 0;
  a[1][0] = 0; a[1][1] = c1; a[1][2] = c2;
  a[2][0] = 0; a[2][1] = -c2; a[2][2] = c1;
  mxm(a, m);
  for (i=0; i<3; i++)
    for (j=0; j<3; j++)
      m[i][j] = a[i][j];
  t.x = r[2].x - r[1].x; t.y = r[2].y - r[1].y; t.z = r[2].z -
r[1].z;
  t = mxb(m, t);
  tmp = (sin(theta[1])*l[2].x - cos(theta[1])*l[2].y);
  *phi1 = atan2(t.z/tmp, t.y/tmp);
  x0.x = r[5].x - r[0].x; x0.y = r[5].y - r[0].y; x0.z = r[5].z -
r[0].z;
  x0 = mxb(m, x0);
  x0.x -= l[1].x;
  x0.y -= l[1].y;
  x0.z -= l[1].z;
  if (fabs(theta[5]) < EPS && fabs(theta[3]) < EPS) {
    c10 = l[3].x*cos(theta[4]);
    c11 = -(cos(theta[2])*l[3].x + sin(theta[2])*l[3].y);
    tmp = sin(theta[2])*l[3].x - cos(theta[2])*l[3].y;
    c10 /= tmp;
    c11 /= tmp;
  } else if (fabs(theta[5]) < EPS && fabs(theta[3]) > EPS) {
    c10 = -l[5].x - l[4].x*cos(theta[4]);
    c11 = -(cos(theta[2])*l[3].x + sin(theta[2])*l[3].y);
    c12 = 1.0/(sin(theta[2])*l[3].x - cos(theta[2])*l[3].y);
  } else if (fabs(theta[3]) > EPS) {
    t.z = 0.0;
    t.x = l[4].x*cos(theta[4]) - l[4].y*sin(theta[4]) + l[5].x;
    t.y = l[4].x*sin(theta[4]) + l[4].y*cos(theta[4]) + l[5].y;
    q12 = vector_length2(t);
    c10 = q12 - vector_length2(l[3]);
    c11 = 2*(cos(theta[2])*l[3].x + sin(theta[2])*l[3].y);
    c12 = -1.0/(2*(sin(theta[2])*l[3].x - cos(theta[2])*l[3].y));
  } else {
    c10 = l[3].x + l[4].x + l[5].x*cos(theta[4]);
    c11 = -cos(theta[2]);
    tmp = sin(theta[2]);
    c10 /= tmp;
    c11 /= tmp;
  }
  c20 = vector_length2(l[5]) - vector_length2(l[4]);
  c21 = 2*(cos(theta[3])*l[4].x + sin(theta[3])*l[4].y);
  c22 = -1.0/(2*(sin(theta[3])*l[4].x - cos(theta[3])*l[4].y));
  fact1 = sin(theta[4])*l[5].x - cos(theta[4])*l[5].y;
  fact2 = l[6].x*cos(theta[5]) + l[6].y*sin(theta[5]);
  u60.x = r[6].x - r[5].x; u60.y = r[6].y - r[5].y; u60.z = r[6].z
- r[5].z;
}
/* This routine returns the F5 function of Doros.
   *n is the number of solutions, which are in f.
*/
void F5(double phi1, double phi2[4], double phi3[4], double
        phi4[4], double f[4], logical valid[4])
{
  int i, j;
  double tmp, c1, c2;
  vector v1, q1, q2, x, y, t, u6;
  double a[3][3], rot1[3][3], rot2[3][3], rot3[3][3], rot4[3][3];
/* determine c1 */
  valid[0] = valid[1] = valid[2] = valid[3] = FALSE;
  flory_rot_matrix(theta[1], phi1, rot1);
  x = bxm(rot1, x0);
  x.x -= l[2].x; x.y -= l[2].y; x.z -= l[2].z;
  v1 = x;
  if (fabs(theta[5]) < EPS && fabs(theta[3]) < EPS) {
    x = bxm(rot1, mxb(m, vector_scale(u60, 1.0)));
    c1 = (c10 + x.x*c11) / sqrt(x.y*x.y + x.z*x.z);
  } else if (fabs(theta[5]) < EPS && fabs(theta[3]) > EPS) {
    x = bxm(m, flory_rot(theta[1], phi1, l[2]));
    r[2].x = x.x + r[1].x; r[2].y = x.y + r[1].y; r[2].z = x.z +
r[1].z;
    t.x = r[5].x - r[2].x; t.y = r[5].y - r[2].y; t.z = r[5].z -
r[2].z;
    x = bxm(rot1, mxb(m, vector_scale(u60,1.0)));
    c1 = c12*(c10 +  vector_dot(t,
         u60)/vector_length(u60)  +  x.x*c11)  /  sqrt(x.y*x.y  +
x.z*x.z);
  } else if (fabs(theta[3]) > EPS) {
    c1 = c12*(c10 - vector_length2(x) + x.x*c11) / sqrt(x.y*x.y +
x.z*x.z);
  } else {
    c1 = (c10 + x.x*c11) / sqrt(x.y*x.y + x.z*x.z);
  }
  /* printf("c1 %lf\n",c1); */
  if (fabs(c1) > 1) return;
/* determine phi2 */
  tmp = asin(c1);
  phi2[0] = phi2[2] = -atan(x.y/x.z);
  if (x.z < 0) phi2[0] = phi2[2] = phi2[0] - PI;
  phi2[0] += tmp;
  phi2[2] += PI - tmp;
  phi2[1] = phi2[0];
  phi2[3] = phi2[2];
  x = v1;
/* determine c2 and phi3 */
```

```
    for (i=0; i<2; i++) {
      y = flory_rotinv(theta[2], phi2[2*i], x);
      y.x -= l[3].x; y.y -= l[3].y; y.z -= l[3].z;
      c2 = c22*(c20 - vector_length2(y) + y.x*c21) / sqrt(y.y*y.y +
y.z*y.z);
      /* printf("c2 %lf\n",c2); */
      if (fabs(c2) <= 1)  {
        tmp = asin(c2);
        phi3[2*i] = phi3[2*i+1] = -atan(y.y/y.z);
        if (y.z < 0) phi3[2*i] = phi3[2*i+1] = phi3[2*i+1] - PI;
        phi3[2*i] += tmp;
        phi3[2*i+1] += PI - tmp;
        valid[2*i] = valid[2*i+1] = TRUE;
      }
    }
    for (i=0; i<4; i++) {
      if (!valid[i]) continue;
/* determine r4 */
      flory_rot_matrix(theta[2], phi2[i], rot2);
      flory_rot_matrix(theta[3], phi3[i], rot3);
      x = mxb(rot3, l[4]);
      x.x += l[3].x; x.y += l[3].y; x.z += l[3].z;
      x = mxb(rot2, x);
      x.x += l[2].x; x.y += l[2].y; x.z += l[2].z;
      x = mxb(rot1, x);
      x.x += l[1].x; x.y += l[1].y; x.z += l[1].z;
      x = bxm(m, x);
      x.x += r[0].x; x.y += r[0].y; x.z += r[0].z;
/* determine F5 */
      if (fabs(theta[5]) < EPS && fabs(theta[3]) < EPS) {
        v1.x = r[6].x - x.x; v1.y = r[6].y - x.y; v1.z = r[6].z -
x.z;
        f[i] = sqrt((l[6].x+l[5].x)*(l[6].x+l[5].x) +
                    l[5].y*l[5].y) - vector_length(v1);
      } else if (fabs(theta[5]) < EPS && fabs(theta[3]) > EPS) {
        x = bxm(m, mxb(rot1, mxb(rot2, mxb(rot3, l[4]))));
        f[i] = vector_dot(x, u60) /
               (vector_length(x)*vector_length(u60)) - cos(theta[4]);
      } else {
        x.x = r[5].x - x.x; x.y = r[5].y - x.y; x.z = r[5].z - x.z;
        x = mxb(m, x);
        x = bxm(rot3, bxm(rot2, bxm(rot1, x)));
        phi4[i] = atan2(x.z/fact1, x.y/fact1);
        u6 = mxb(m, u60);
        x.x = 1.0; x.y = 0; x.z = 0;
        f[i] = vector_dot(u6, mxb(rot1, mxb(rot2, mxb(rot3,
                             flory_rot(theta[4],  phi4[i],  x))))) -
fact2;
    }
  }
}

*****************************************************************
              GEOMETRY/ROTATION ROUTINES - PEPTIDE7.C
*****************************************************************

/*                        The geometry routines
*/
#include "peptide.h"
/* This routine rotates the vector a about n by theta
(counterclockwise is +)
  r' = r cos(theta) + n(n.r)(1-cos(theta)) + nxr sin(theta)
*/
vector vector_rotate(vector a, vector n, double cos_theta, double
sin_theta)
{
  double fact;
  vector ret, v;
  fact = (n.x*a.x + n.y*a.y + n.z*a.z) * (1.0 - cos_theta);
  v = vector_cross(n,a);
  ret.x = a.x*cos_theta + n.x*fact + v.x*sin_theta;
  ret.y = a.y*cos_theta + n.y*fact + v.y*sin_theta;
  ret.z = a.z*cos_theta + n.z*fact + v.z*sin_theta;
  return(ret);
}
/* This routine returns main-chain b0
   i=0 noncyclic case should never happen--it won't be right
*/
```

```
vector get_main_b0(atom_list *atom, regrowth *main, int i)
{
  vector x, y;
  if (main[i].prev == NULL) {
    x.x = x.y = 0.0;
    x.z = 1.0;
    return(x);
  }
   x    =    atom[main[i].unit->list_num    +
main[i].unit->head.atom_num].position;
                                                          y    =
atom[main[i].prev->bond[main[i].prev->n_bonds-1]->tail.atom_num +
          main[i].prev->list_num].position;
  x.x -= y.x;
  x.y -= y.y;
  x.z -= y.z;
  return(vector_scale(x, 1.0));
}
/* This routine returns main-chain p0
   i=0 noncyclic case should never happen--it won't be right
*/
vector get_main_p0(atom_list *atom, regrowth *main, int i)
{
  vector x;
  if (main[i].prev == NULL) {
    x.x = x.y = x.z = 0.0;
    return(x);
  }
                                                          x    =
atom[main[i].prev->bond[main[i].prev->n_bonds-1]->tail.atom_num +
          main[i].prev->list_num].position;
  return(x);
}
/* This routine returns side-chain b0 */
vector get_side_b0(atom_list *atom, regrowth *side, int i)
{
  vector x, y;
   x    =    atom[side[i].unit->list_num    +
side[i].unit->head.atom_num].position;
```

```
   y    =    a t o m [ s i d e [ i ] . p r e v - > l i s t _ n u m    +
side[i].prev->head.atom_num].position;
  x.x -= y.x;
  x.y -= y.y;
  x.z -= y.z;
  return(vector_scale(x, 1.0));
}
/* This routine returns side-chain p0 */
vector get_side_p0(atom_list *atom, regrowth *side, int i)
{
  vector x;
   x    =    a t o m [ s i d e [ i ] . p r e v - > l i s t _ n u m    +
side[i].prev->head.atom_num].position;
  return(x);
}
/* This routine gives the Flory rotation matrix
*/
void flory_rot_matrix(double theta, double phi, double m[3][3])
{
  double cost, sint, cosp, sinp;
  cost = cos(theta); sint = sin(theta);
  cosp = cos(phi); sinp = sin(phi);
  m[0][0] = cost;
  m[0][1] = sint;
  m[0][2] = 0.0;
  m[1][0] = sint*cosp;
  m[1][1] = -cost*cosp;
  m[1][2] = sinp;
  m[2][0] = sint*sinp;
  m[2][1] = -cost*sinp;
  m[2][2] = -cosp;
}
/* This routine does the Flory rotation
*/
vector flory_rot(double theta, double phi, vector a)
{
  vector t;
  double cost, sint, cosp, sinp, tmp;
  cost = cos(theta); sint = sin(theta);
  cosp = cos(phi); sinp = sin(phi);
  tmp = sint*a.x - cost*a.y;
  t.x = cost*a.x + sint*a.y;
  t.y = cosp*tmp + sinp*a.z;
  t.z = sinp*tmp - cosp*a.z;
  return(t);
}
/* This routine does the inverse Flory rotation
*/
vector flory_rotinv(double theta, double phi, vector a)
{
  vector t;
  double cost, sint, cosp, sinp, tmp;
  cost = cos(theta); sint = sin(theta);
  cosp = cos(phi); sinp = sin(phi);
  tmp = cosp*a.y + sinp*a.z;
  t.x = cost*a.x + sint*tmp;
  t.y = sint*a.x - cost*tmp;
  t.z = sinp*a.y - cosp*a.z;
  return(t);
}
/* This routine constructs the lab transformation to go from l to
r
*/
void flory_lab(double m[3][3], vector r, vector l)
{
  double sin_theta, cos_theta;
  vector n;
  r = vector_scale(r, 1.0);
  l = vector_scale(l, 1.0);
  n = vector_cross(l,r);
  cos_theta = vector_dot(l,r);
  sin_theta = vector_length(n);
  if (sin_theta < EPS) {
    n.x = 1.0;
  } else {
    n.x /= sin_theta;
    n.y /= sin_theta;
    n.z /= sin_theta;
  }
  m[0][0] = cos_theta + n.x*n.x*(1.0-cos_theta)                ;
  m[0][1] =             n.x*n.y*(1.0-cos_theta) - sin_theta*n.z;
  m[0][2] =             n.x*n.z*(1.0-cos_theta) + sin_theta*n.y;
  m[1][0] =             n.y*n.x*(1.0-cos_theta) + sin_theta*n.z;
  m[1][1] = cos_theta + n.y*n.y*(1.0-cos_theta)                ;
  m[1][2] =             n.y*n.z*(1.0-cos_theta) - sin_theta*n.x;
  m[2][0] =             n.z*n.x*(1.0-cos_theta) - sin_theta*n.y;
  m[2][1] =             n.z*n.y*(1.0-cos_theta) + sin_theta*n.x;
  m[2][2] = cos_theta + n.z*n.z*(1.0-cos_theta)                ;
}
/* This routine constructs the inverse lab transformation
*/
void flory_labinv(double m[3][3], vector r, vector l)
{
  double sin_theta, cos_theta;
  vector n;
  r = vector_scale(r, 1.0);
  l = vector_scale(l, 1.0);
  n = vector_cross(l,r);
  cos_theta = vector_dot(l,r);
  sin_theta = vector_length(n);
  if (sin_theta < EPS) {
    n.x = 1.0;
  } else {
    n.x /= sin_theta;
    n.y /= sin_theta;
    n.z /= sin_theta;
  }
  m[0][0] = cos_theta + n.x*n.x*(1.0-cos_theta)                ;
  m[1][0] =             n.x*n.y*(1.0-cos_theta) - sin_theta*n.z;
  m[2][0] =             n.x*n.z*(1.0-cos_theta) + sin_theta*n.y;
  m[0][1] =             n.y*n.x*(1.0-cos_theta) + sin_theta*n.z;
  m[1][1] = cos_theta + n.y*n.y*(1.0-cos_theta)                ;
  m[2][1] =             n.y*n.z*(1.0-cos_theta) - sin_theta*n.x;
  m[0][2] =             n.z*n.x*(1.0-cos_theta) - sin_theta*n.y;
  m[1][2] =             n.z*n.y*(1.0-cos_theta) + sin_theta*n.x;
  m[2][2] = cos_th ta + n.z*n.z*(1.0-cos_theta)                ;
}
```

```
/* This routine returns a vector cross product
*/
vector vector_cross(vector a, vector b)
{
  vector ret;
  ret.x = a.y*b.z - a.z*b.y;
  ret.y = a.z*b.x - a.x*b.z;
  ret.z = a.x*b.y - a.y*b.x;
  return(ret);
}
/* This function scales the vector v so that |v| = r
*/
vector vector_scale(vector v, double r)
{
  double ftmp;
  ftmp = sqrt(v.x*v.x + v.y*v.y + v.z*v.z);
  v.x *= r/ftmp;
  v.y *= r/ftmp;
  v.z *= r/ftmp;
  return(v);
}
/* This routine returns mxn in m
*/
void mxm(double m[3][3], double n[3][3])
{
  int i,j,k;
  double a[3][3];
  for (i=0; i<3; i++)
    for (j=0; j<3; j++) {
      a[i][j] = 0.0;
      for (k=0; k<3; k++) a[i][j] += m[i][k]*n[k][j];
    }
  for (i=0; i<3; i++)
    for (j=0; j<3; j++)
      m[i][j] = a[i][j];
}
/* This routine deturns det(m), where m is 5x5
*/
double det5(doubl  m[5][5])
```

```
{
  int i,j,k;
  double a[5][5], fact;
  for (i=0; i<5; i++)
    for (j=0; j<5; j++)
      a[i][j] = m[i][j];
  for (i=0; i<4; i++) {
    for (k=i+1; k<5; k++) {
      fact = a[k][i] / a[i][i];
      for (j=i; j<5; j++) a[k][j] -= fact*a[i][j];
    }
  }
  return(a[0][0]*a[1][1]*a[2][2]*a[3][3]*a[4][4]);
}
/* This routine returns det(m), where m is 3x3
*/
double det(double m[3][3])
{
  return(m[0][0]*m[1][1]*m[2][2] + m[0][1]*m[1][2]*m[2][0] +
         m[0][2]*m[1][0]*m[2][1] - m[2][0]*m[1][1]*m[0][2] -
         m[1][0]*m[0][1]*m[2][2] - m[0][0]*m[2][1]*m[1][2]);
}
/* This routine returns Mb
*/
vector mxb(double m[3][3], vector b)
{
  vector t;

  t.x = m[0][0]*b.x + m[0][1]*b.y + m[0][2]*b.z;
  t.y = m[1][0]*b.x + m[1][1]*b.y + m[1][2]*b.z;
  t.z = m[2][0]*b.x + m[2][1]*b.y + m[2][2]*b.z;
  return(t);
}
/* This routine returns Mb
*/
vector bxm(double m[3][3], vector b)
{
  vector t;
  t.x = m[0][0]*b.x + m[1][0]*b.y + m[2][0]*b.z;
  t.y = m[0][1]*b.x + m[1][1]*b.y + m[2][1]*b.z;
  t.z = m[0][2]*b.x + m[1][2]*b.y + m[2][2]*b.z;
  return(t);
}
/* This routine returns b1.b2
*/
double vector_dot(vector b1, vector b2)
{
  return(b1.x*b2.x + b1.y*b2.y + b1.z*b2.z);
}
/* This routine returns |v|
*/
double vector_length(vector v)
{
  return(sqrt(v.x*v.x + v.y*v.y + v.z*v.z));
}
/* This routine returns |v|^2
*/
double vector_length2(vector v)
{
  return(v.x*v.x + v.y*v.y + v.z*v.z);
}

*****************************************************************
              RANDOM NUMBER GENERATOR - RANDOM.C
*****************************************************************

/*
  This is the pseudo-random number library.
*/
#include <time.h>
/*
  This function returns a random number in [0,1).
  It uses a linear-congruential method.
  ran(0.0) initializes the random number seed with a time dependant
valu
    and returns the value of the s ed that the generator
recognizes.
```

```
  ran(1.0) returns the next number in the random sequence.
  Other arguments initialize the seed with the user-supplied value.
    Initializing the generator with a seed from the sequence, will
cause the
    subsequent ran(1.0) to generate the next value of the sequence.
    This is usefull, for example, to shut down and start up the
generator
    without a loss of continuity in the sequence.
  Values r 1 or < 0 are not recommended.
  It has a period of M.
*/
double ran(double dummy)
{
  static long int ix;
  double rm = 566927.0, rm2 = 1.0/rm;
  long int k = 5701, j = 3621, m = 566927, tmp;
     /* make sure parameters not too far off */
  if (dummy > 2.0) dummy = 2.0;
  if (dummy < -2.0) dummy = -2.0;
  if (dummy != 1.0)
  {
      if ((tmp = dummy*rm) < m)
        ix = tmp;
      else
        ix = m-1;
      if (ix < 0)
        ix = 0;
  } else
    ix = (j*ix + k) % m;
  return(ix * rm2);
}
/*
  This function returns a pseudo-random number in (0,1).
  This is a more robust pseudo-random number generator than a
simple linear-
    congruential gererator is.
  It uses three linear congruential generators to get one random
number.
  ran2(0.0) initializes the generator with time-dependent values.
  ran2(1.0) returns a pseudo-random number.
  Other arguments are used as an initializing seed.
  Arguments r 1 or s 0 are ill-advised.
  It has a period of (m1-1)(m2-1)(m3-1)/4.
*/
double ran2(double dummy)
{
  double f1=1.0/30269.0 ,f2=1.0/30307.0, f3=1.0/30323.0, tmp;
  int m1=30269, m2=30307, m3=30323, seed, itmp;
  static x,y,z;
     /* make sure parameters not too far off */
  if (dummy > 1.1) dummy = 1.1;
  if (dummy < -1.1) dummy = -1.1;
  if (dummy != 1.0)
  {
                 /* initialize with user's seed value */
     if ((itmp = dummy*m1) < m1)
       seed = itmp;
     else
       seed = m1-1;
   if (seed < 1) seed = 1;
                              /* initialize first generator */
   x = seed;
                              /* initialize second generator */
   y = 172 * (x % 176)  -  35 * (x/176);
   if (y < 0) y += m2;
                              /* initialize third generator */
   z = 170 * (y % 178)  -  63 * (y/178);
   if (z < 0) z += m3;
 }
                                      /* first generator */
 x = 171 * (x % 177)  -  2 * (x/177);
 if (x < 0) x += m1;
                                      /* second generator */
 y = 172 * (y % 176)  -  35 * (y/176);
 if (y < 0) y += m2;
                                      /* third generator */
 z = 170 * (z % 178)  -  63 * (z/178);
 if (z < 0) z += m3;
```

```
                                              /* amalgamated result */
  itmp = tmp = x*f1 + y*f2 + z*f3;
  return(tmp - itmp);
}

*****************************************************************
*****************************************************************
                        C INCLUDE FILES
*****************************************************************
*****************************************************************

*****************************************************************
               GLOBAL VARIABLE TYPES - PEP_TYPE.H
*****************************************************************

/* Global types used in the program */
typedef enum {FALSE, TRUE} logical;
typedef enum {BAD, G, A, V, L, I, S, T, D, E, N, Q, K, H, R, F, Y,
W, C, M, P}
      acid_label;
typedef enum {UNKNOWN, nonCunit, Cunit} unit_label;
typedef struct {
                double x,y,z;
              } vector;
typedef struct {
                vector axis;
                int atom_num;
                int bond[MAX_BONDS];
              } connector;
typedef struct bond_struct {
                connector tail;
                struct rigid_unit_struct *next;
              } bond_type;
typedef char *string;

typedef struct {
                char name[NAME_LENGTH];
                char type[NAME_LENGTH];
                double charge, ri, ei;
```

```
                    vector position;
                    acid_label residue;
                    int residue_num;
                  } atom_info;
typedef struct rigid_unit_struct {
                    unit_label type;
                    connector head;
                    int list_num;
                    int n_bonds;
                    bond_type **bond;
                    int n_atoms;
                    atom_info *atom;
                  } rigid_unit;
typedef struct {
                    atom_info *p;
                    vector position;
                  } atom_list;
typedef struct {
                    char type1[NAME_LENGTH], type2[NAME_LENGTH],
                         type3[NAME_LENGTH], type4[NAME_LENGTH];
                    double v0[3], phi0[3];
                  } torsion_data;
typedef struct torsion_list_struct {
                    int num[4];
                    torsion_data *p;
                    int degen;
                    struct torsion_list_struct *next;
                  } torsion_list;
typedef struct {
                    char type[NAME_LENGTH];
                    double ri, ei;
                  } lj_data;
typedef struct {
                    char type1[NAME_LENGTH], type2[NAME_LENGTH];
                    double a, b;
                  } hbond_data;
typedef struct hbond_list_struct {
                    int num[2];
                    hbond_data *p;
                  struct hbond_list_struct *next;
                } hbond_list;
typedef struct {
                  rigid_unit *unit, *prev;
                } regrowth;

*****************************************************************
                  GLOBAL VARIABLES - PEP_VAR.H
*****************************************************************

/* Global variables used in the program */
#if defined(MAIN)
#define EXT extern
#else
#define EXT
#endif
EXT torsion_data **torsion_data_list;
EXT lj_data **lj_data_list;
EXT hbond_data **hbond_data_list;
#undef EXT

*****************************************************************
                  GLOBAL FUNCTIONS - PEPTIDE.H
*****************************************************************

/* Include files needed by peptide code */
#include <stdio.h>
#include <float.h>
#include <math.h>
#include <fcntl.h>
#include <stdio.h>
#include <memory.h>
#include <malloc.h>
#include <string.h>
#include <search.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <time.h>
```

```
#include <varargs.h>
/* global constants */
#define BETA 1.6886683 /* kB T at 298K */
#define MAX_BONDS 8
#define PI 3.1415927
#define EPS 1.0E-9
#define NAME_LENGTH 10
#define KMAX 100
#define MAX_ROOTS 100
#define DPHI .01
/* global macros */
#define INTERVAL(a,n1,n2) ((a) >= (n1) && (a) < (n2))
/* Include files relevant to this program */
#include "pep_type.h"
#include "pep_var.h"
/* random.c */
double ran(double dummy);
double ran2(double dummy);
/* peptide1.c */
void out_of_memory(void);
void get_sequence(string **sequence, int *n_peptides);
rigid_unit *read_peptide_data(string sequence, int *n_atoms_total,
                              int *max_atoms_per_unit);
rigid_unit   *read_unit(string   file,   acid_label   label,   int
residue_num,
                     int *n_atoms_total, int *max_atoms_per_unit);
void couple_unit(rigid_unit *unit1, rigid_unit *unit2);
rigid_unit    *modify_cystine_ends(rigid_unit    *unit,    int
n_amino_acids,
                               int *n_atoms_total);
void  get_main_side(rigid_unit  *unit,  regrowth  *main,  regrowth
*side,
                   int *n_main, int *n_side);
void read_torsion_data(void);
void read_lj_data(void);
void read_hbond_data(void);
void write_car_file(int n_amino_acids, int n_atoms_total, atom_list
*atom,
                    string file);
string getline(string line, int len, FILE *fp);
void strip(string string);
void decomma(string string);
void capitalize(string s);
void amino_acid_code_3(acid_label label, string code_3);
void amino_acid_code_1(acid_label label, char code_1);
acid_label amino_acid_code(char code_1);
/* peptide2.c */
void    initialize_connection_table(int    **bond_table,    int
n_atoms_total);
void make_connection_table(int **bond_table, int *table_num,
                           rigid_unit *unit, rigid_unit *start);
void add_connection(int **bond_table, int i1, int i2);
void print_connection_table(int **bond_table, int n_atoms_total);
void   get_torsions(torsion_list   **p,   int   **bond_table,   int
*table_num,
                    atom_list *atom, rigid_unit *unit, rigid_unit
*start);
torsion_list *add_torsion(int **bond_table, atom_list *atom, int
i, int j,
                          int k, int l);
logical  lookup_torsion_data(string type1,  string type2,  string
type3,
                             string type4, torsion_data **p);
void print_torsions(torsion_list *list, atom_list *atom);
double torsion(vector p1, vector p2, vector p3, vector p4);
void assign_lj_parameters(rigid_unit *unit, rigid_unit *start);
logical lookup_lj_data(string type, double *ri, double *ei);
logical lookup_lj_data(string type, double *ri, double *ei);
void get_hbonds(hbond_list **list, atom_list *atom, int n_atoms);
logical lookup_hbond_data(string type1, string type2, hbond_data
**p);
void print_hbonds(hbond_list *l, atom_list *atom);
void   assign_atom_pointers(int   *list_num,   rigid_unit   *unit,
rigid_unit *start,
                          atom_list *atom);
/* peptide3.c */
void old_unit(int *list_num, int n0, int n1, int n2, double
*logrosen,
             rigid_unit *unit, rigid_unit *start, torsion_list *t,
             hbond_list  *l,  atom_list  *atom,  vector  *twig[],
vector p0,
             vector b0);
void do_unit(int  *list_num,  int  n0,  int  n1,  int  n2,  double
*logrosen,
             rigid_unit *unit, rigid_unit *start, torsion_list *t,
             hbond_list *l, atom_list *atom, vector *twig[], vector
p0,
             vector b0, double *e);
void do_backbone_f(int i, int n_main, int n_atoms_total,
                   double *logrosen,
                   regrowth *main, regrowth *side,
                   torsion_list *t, hbond_list *l,
                   atom_list *atom, vector *twig[],
                   double *e, logical new);
void do_backbone_f_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list *atom, atom_info *atom_tmp,
                         vector *twig[],
                         double *e, logical new);
void do_backbone_b(int i, int n_main, int n_atoms_total,
                   double *logrosen,
                   regrowth *main, regrowth *side,
                   torsion_list *t, hbond_list *l,
                   atom_list *atom, vector *twig[],
                   double *e, logical new);
void do_backbone_b_rigid(int i, int n_main, int n_atoms_total,
                         double *logrosen,
                         regrowth *main, regrowth *side,
                         torsion_list *t, hbond_list *l,
                         atom_list  *atom,  atom_info  *atom_tmp,
vector *twig[],
                         double *e, logical new);
void do_unit_sub(int  *list_num,  int  n0,  int  n1,  int  n2,  double
*logrosen,
                rigid_unit *unit, torsion_list *t, hbond_list *l,
                    atom_list *atom, vector *twig[], vector p1, vector
b1,
                    vector p0, vector b0, double *e, vector
p[MAX_BONDS],
                    vector b[MAX_BONDS], logical new);
void add_rigid_unit(rigid_unit *unit, vector *pos,
                      vector p1, vector b1, vector p0,
                      vector b0, vector point[MAX_BONDS],
                      vector bond[MAX_BONDS],
                      double cos_theta2, double sin_theta2);
vector align(vector p, vector r0, vector r1, vector n, double
cos_theta,
               double sin_theta, vector n2, double cos_theta2, double
sin_theta2);
/* peptide4.c */
double delta_energy(torsion_list *t, hbond_list *l, atom_list
*atom,
                      vector *twig, int n_atoms, int n0, int n1, int
n2,
                      int n_twig);
double energy(torsion_list *t, hbond_list *l, atom_list *atom,
               int n_atoms_total);
double d_nonbond_energy(torsion_list *t, atom_list *atom, vector
*twig,
                          int n_atoms, int n0, int n1, int n2, int
n_twig);
double nonbond_energy(torsion_list *t, atom_list *atom, int
n_atoms_total);
double d_hbond_energy(hbond_list *l, atom_list *atom, vector *twig,
                       int n_atoms, int n0, int n1, int n2, int
n_twig);
double hbond_energy(hbond_list *l, atom_list *atom);
double d_torsion_energy(torsion_list *t, atom_list *atom, vector
*twig,
                          int n_atoms, int n0, int n1, int n2, int
n_twig);
double torsion_energy(torsion_list *t, atom_list *atom);
/* peptide5.c */
void do_mc(rigid_unit *unit, torsion_list *t, hbond_list *l,
           atom_list *atom, atom_list *atom2, atom_info *atom_tmp,
           vector *twig[], regrowth *main, regrowth *side,
           int n_amino_acids, int n_atoms_total, int n_main, int
n_side,
           logical cyclic);
void read_restart(atom_list *atom, int n_atoms_total);
void read_cycle(torsion_list *t, hbond_list *l,
                atom_list *atom, regrowth *main, regrowth *side,
                vector  *twig[],  int  n_main,  int  n_side,  int
n_atoms_total);
void regrow_main(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2,
                 atom_info *atom_tmp, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_main, int n_atoms_total, double *e);
void regrow_side(torsion_list *t, hbond_list *l,
                 atom_list *atom, atom_list *atom2, vector *twig[],
                 regrowth *main, regrowth *side,
                 int n_side, int n_atoms_total, double *e);
/* peptide6.c */
void rotate_main(atom_list *atom, atom_list *atom2, vector *twig[],
                 regrowth *main, regrowth *side, torsion_list *t,
                 hbond_list  *l,  int  n_main,  int  n_atoms_total,
double *e);
void get_rot_params(atom_list *atom, regrowth *main, int i0, int
n_main);
void get_rot_rosenbluth(atom_list *atom, atom_list *atom2,
                         vector *twig[], regrowth *main,
                        torsion_list *t, hbond_list *l, int i0, int
n_main,
                        int n_atoms_total, int *n, int *j, double
*logrosen,
                        double *e);
double jac(vector r[7]);
vector  rotate_r1(atom_list  *atom,  regrowth  *main,  int  i0,  int
n_main);
void get_r(double phi1, double phi2, double phi3, double phi4,
double phi5);
void do_rotation(atom_list *atom, vector *twig, regrowth *main, int
i0,
                int n_main, int n_atoms_total);
void get_phi1(double phi[MAX_ROOTS][6], int *n);
void get_root(double x0, double x1, double *p1, double *p2, double
*p3,
              double *p4, double *p5, int n);
void F5init(vector q2, double *phi1);
void F5(double phi1, double phi2[4], double phi3[4], double
phi4[4],
        double phi5[4], double f[4], logical valid[4]);
/* peptide7.c */
vector vector_rotate(vector a, vector n, double cos_theta, double
sin_theta);
vector get_main_b0(atom_list *atom, regrowth *main, int i);
vector get_main_p0(atom_list *atom, regrowth *main, int i);
vector get_side_b0(atom_list *atom, regrowth *side, int i);
vector get_side_p0(atom_list *atom, regrowth *side, int i);
void flory_rot_matrix(double theta, double phi, double m[3][3]);
vector flory_rot(double theta, double phi, vector a);
vector flory_rotinv(double theta, double phi, vector a);
void flory_lab(double m[3][3], vector r, vector l);
void flory_labinv(double m[3][3], vector r, vector l);
vector vector_cross(vector a, vector b);
vector vector_scale(vector v, double r);
void mxm(double m[3][3], double n[3][3]);
double det5(double m[5][5]);
double det(double m[3][3]);
vector mxb(double m[3][3], vector b);
vector bxm(double m[3][3], vector b);
double vector_dot(vector b1, vector b2);
double vector_length(vector v);
double vector_length2(vector v);

*****************************************************************
*****************************************************************
             DATA FILES DEFINING GEOMETRIC STRUCTURE
*****************************************************************
*****************************************************************
```

```
*****************************************************************
              DATA FILE FOR UNIT A - UNITA.DAT
*****************************************************************

! data file for rigid unit A--the NH2 terminus
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
N         0.039039567   -0.028048204    0.000005808 ALAn 1      NT
    N  -0.463
HN1      -0.294595420    0.946419656    0.000007165 ALAn 1       H
     H   0.126
HN2      -0.309849501   -0.509882152   -0.840834498 ALAn 1       H
     H   0.126
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond--doesn't mean anything, but
must not be 1
0 0 .00000001!beginning of incoming bond -- just an overall
displacement
1 !bond out from this unit
-1 !don't know which unit this bond goes to
0 1 2 -1 -1 !beginning of outgoing backbone bond
1.498959541   -0.043336947   -0.000000042 !ending of outoing bond

*****************************************************************
              DATA FILE FOR UNIT B - UNITB.DAT
*****************************************************************

! data file for rigid unit B--the CH alpha carbon unit
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
2 !atoms in this rigid unit
CA        4.047343731    2.755753756   -0.000011837 ALA  2      CT
    C   0.035
HA        3.779272556    3.294512749   -0.928205431 ALA  2      HC
    H   0.032
```

```
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1!ending of incoming backbone bond
3.370934725     1.461895347    -0.000009674 !beginning of incoming
backbone bond
2 !bonds out from this unit
-1 !don't know which unit this bond goes to
0 1 -1 -1 -1 !beginning of outgoing side-chain bond
3.538550615      3.547572851      1.217100978 !ending of outgoin
side-chain bond
-1 !don't know which unit this bond goes to
0 1 -1 -1 -1!beginning of outgoing backbone bond
5.547336102      2.582198620    -0.000015057 !ending of outgoing
backbone bond

*****************************************************************
                 DATA FILE FOR UNIT C - UNITC.DAT
*****************************************************************

! data file for rigid unit C--the OCNH amide bond unit
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
4 !atoms in this rigid unit
C          2.054825068    1.360626340    0.000001071 ALAn 1      C
     C   0.616
O          1.320880890    2.356072187    0.011419594 ALAn 1      O
     O  -0.504
N          3.370934725    1.461895347   -0.000009674 ALA  2      N
     N  -0.463
HN         3.917454243    0.530382395   -0.000003380 ALA  2      H
     H   0.252
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming main-chain bond
1.498959541   -0.043336947   -0.000000042 !beginning of incoming
main-chain bond
1 !bond out from this unit
-1 !don't know which unit this bond goes to
```

```
2 0 3 -1 -1 !beginning of outgoing main-chain bond
4.047343731      2.755753756     -0.000011837 !ending of outging
main-chain bond

*******************************************************************
                 DATA FILE FOR UNIT D - UNITD.DAT
*******************************************************************

! data file for rigid unit D--the HCO terminus
1 !rigid unit in this structure
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
C         8.274295807    5.082911491   -0.000008575 ALAN 3       C
     C   0.616
HC        9.361082077    5.166533947   -0.000010758 ALAN 3      HC
    H   0.000
O         7.540351391    6.078356743    0.011415332 ALAN 3       O
     O  -0.504
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming main-chain bond
7.718430996     3.678948641    -0.000013665 !beginning of incoming
main-chain bond
0 !bonds out from this unit

***************************************************************
                 DATA FILE FOR ALANINE - A.DAT
***************************************************************

! The side-chain structure file for Alanine
1 !rigid unit in side-chain
! ATOM INFORMATION
! rigid unit 0
4 !atoms in this rigid unit
CB        3.178086281    3.790203094    1.217109203 ALA  2
CT       C  -0.098
HB1       3.502361059    4.845792770    1.274110079 ALA  2
HC       H   0.038
```

```
HB2       2.072028160    3.800241470    1.180677295 ALA  2
HC       H   0.038
HB3       3.465983868    3.309211969    2.172164917 ALA  2
HC       H   0.038
! BOND INFORMATION
! rigid unit 0
0 1 2 3 -1 !ending of incoming bond for unit 0 and nn
 3.783586502   3.069634676   -0.000003090 !beginning of bond for
unit 0
0 !bonds out from rigid unit 0

************************************************************
                DATA FILE FOR CYSTEINE - C.DAT
*************************************************************

! The side-chain structure file for Cysteine
! Do not modify the atom order in this file
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.185384274    3.813543320    1.210355163 CYSH 2
CT       C  -0.060
HB1       2.082855701    3.742515087    1.217666388 CYSH 2
HC       H   0.038
HB2       3.528102398    3.371057510    2.168041706 CYSH 2
HC       H   0.038
! rigid unit 1
4 !atoms in this rigid unit
SG        3.628824234    5.564641953    1.168115854 CYSH 2
SH       S   0.827
LG1       2.774378061    6.223292828    1.382826447 CYSH 2
LP       L  -0.481
LG2       4.018448353    5.879447937    0.188784361 CYSH 2
LP       L  -0.481
HG        4.543437004    5.521058083    2.133599997 CYSH 2
HS       H   0.135
! BOND INFORMATION
! rigid unit 0
```

```
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634914   -0.000003354 !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 3.628824234    5.564641953    1.168115854  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 3 -1 !ending of incoming bond for unit 1 and nn
3.185384274   3.813543320   1.210355163 !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

*********************************************************
                DATA FILE FOR ASPARTATE - D.DAT
*********************************************************

! The side-chain structure file for Aspartate
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.195193052    3.859569550    1.198083878 ASP  2
CT       C  -0.398
HB1       2.099623203    3.734851122    1.256908774 ASP  2
HC       H   0.071
HB2       3.574837923    3.424842119    2.144523859 ASP  2
HC       H   0.071
! rigid unit 1
3 !atoms in this rigid unit
CG        3.488366127    5.366341114    1.240691185 ASP  2
C        C   0.714
OD1       3.752036572    5.965095997    2.273211718 ASP  2
O2       O  -0.721
OD2       3.445515871    5.949848175    0.005213364 ASP  2
O2       O  -0.721
! BOND INFORMATION
! rigid unit 0
```

```
0 1 2 -1 -1 ! nding of incoming bond for unit 0 and nn
3.783586502   3.069634438   -0.000003352  !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 3.488366127    5.366341114    1.240691185  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.195193052   3.859569550   1.198083878  !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

********************************************************
              DATA FILE FOR GLUTAMINE - E.DAT
********************************************************

! The side-chain structure file for Glutamine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB         3.210191727    3.806770086    1.242457986 GLU  2
CT       C  -0.184
HB1        3.453276873    4.884052753    1.160096049 GLU  2
HC       H   0.092
HB2        2.103818655    3.775332928    1.193925381 GLU  2
HC       H   0.092
! rigid unit 1
3 !atoms in this rigid unit
CG         3.670672178    3.303917646    2.650651217 GLU  2
CT       C  -0.398
HG1        3.495624304    2.214699984    2.732162237 GLU  2
HC       H   0.071
HG2        4.766538143    3.410970449    2.754028797 GLU  2
HC       H   0.071
! rigid unit 2
3 !atoms in this rigid unit
```

```
CD        3.044564962    3.944746017    3.891577959 GLU  2
C         C   0.714
OE1       3.318646908    3.594962835    5.031950951 GLU  2
O2        O  -0.721
OE2       2.157183647    4.937835217    3.607111931 GLU  2
O2        O  -0.721
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
 3.783586502    3.069634438   -0.000003351  !beginning of bond
for unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.670672178    3.303917646    2.650651217   !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.210191727    3.806770086    1.242457986 !beginning of bond for
unit 1
1 !bonds out from rigid unit 1
2 !unit 1 is bonded to unit 2
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 3.044564962    3.944746017    3.891577959   !ending of outgoing
bond for unit 1
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.670672178    3.303917646    2.650651217 !beginning of bond for
unit 2
0 !bonds out from rigid unit 2
```

****

DATA FILE FOR PHENYLALANINE - F.DAT

****

```
! The side-chain structure file for Phenylalanine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
```

```
3 !atoms in this rigid unit
CB        3.271046400    3.829343796    1.261018753 PHE  2
CT       C  -0.100
HB1       3.711064339    3.375446320    2.172759056 PHE  2
HC       H   0.108
HB2       3.680548668    4.858696938    1.261503935 PHE  2
HC       H   0.108
! rigid unit 1
11 !atoms in this rigid unit
CG        1.746863961    3.913921356    1.435816050 PHE  2
CA       C  -0.100
CD1       1.070973635    2.894981861    2.116770267 PHE  2
CA       C  -0.150
HD1       1.621361971    2.061387062    2.533305407 PHE  2
HC       H   0.150
CD2       1.019180536    4.963639259    0.869901121 PHE  2
CA       C  -0.150
HD2       1.528048277    5.750367641    0.331381440 PHE  2
HC       H   0.150
CE1      -0.315989435    2.915796280    2.214086056 PHE  2
CA       C  -0.150
HE1      -0.830357015    2.108316422    2.715482712 PHE  2
HC       H   0.150
CE2      -0.369023502    4.989082813    0.977358818 PHE  2
CA       C  -0.150
HE2      -0.928361893    5.798536777    0.531342983 PHE  2
HC       H   0.150
CZ       -1.036266327    3.964326382    1.646436572 PHE  2
CA       C  -0.150
HZ       -2.113304853    3.975853443    1.718335271 PHE  2
HC       H   0.150
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586264    3.069634914   -0.000003353 !beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
1.746863961    3.913921356    1.435816050 !ending of outgoing
```

```
bond
! rigid unit 1
0 1 3 -1 -1 !ending of incoming bond and nn
3.271046400    3.829343796    1.261018753 !beginning of bond
0 !bonds out

**********************************************************
                DATA FILE FOR GLYCINE - G.DAT
************************************************************

! The side-chain structure file for Glycine
1 !rigid unit in side-chain
! ATOM INFORMATION
! rigid unit 0
1 !atom in this rigid unit
HA2       2.054570675    -0.518772364    -0.887896836 GLYN 1
HC       H   0.032
! BOND INFORMATION
! rigid unit 0
0 -1 -1 -1 -1 !ending of incoming bond for unit 0 and nn
1.612465143   -0.031237146   -0.000000015 !beginning of incoming
bond for unit 0
0 !bonds out from rigid unit 0

**********************************************************
               DATA FILE FOR HISTIDINE - H.DAT
*********************************************************

! The side-chain structure file for Histidine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.239844084    3.731920242    1.277127385 HIS  2
CT       C  -0.098
HB1       2.644425392    3.025787830    1.893024564 HIS  2
HC       H   0.038
HB2       4.064783096    4.071127415    1.934927344 HIS  2
HC       H   0.038
```

```
! rigid unit 1
8 !atoms in this rigid unit
CG        2.370461226    4.918142319    0.978080690 HIS  2
CC        C   0.251
ND1       2.062596560    5.403582573   -0.290515751 HIS  2
NB        N  -0.502
CE1       1.272076607    6.440367222    0.045922592 HIS  2
CR        C   0.241
NE2       1.048720956    6.674089432    1.367565274 HIS  2
NA        N  -0.146
CD2       1.767608762    5.675839901    1.972463250 HIS  2
CW        C  -0.184
HE1       0.858503580    7.036557198   -0.757577479 HIS  2
HC        H   0.036
HE2       0.480951071    7.411210537    1.809884906 HIS  2
H         H   0.228
HD2       1.867301583    5.485908508    3.037219763 HIS  2
HC        H   0.114
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634438   -0.000003353 !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
2.370461226    4.918142319    0.978080690 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 4 -1 -1 !ending of incoming bond for unit 1 and nn
3.222899199   3.830397844   1.236912012 !beginning of bond for
unit 1
0 !bonds out from rigid unit 1
```

```
*************************************************************
                DATA FILE FOR ISOLEUCINE - I.DAT
*****************************************************************

! The side-chain structure file for Isoleucine
```

```
4 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
2 !atoms in this rigid unit
CB        3.184130907    3.905461311    1.203313947 ILE  2
CT        C  -0.012
HB        3.579479933    3.448693275    2.135145664 ILE  2
HC        H   0.022
! rigid unit 1
4 !atoms in this rigid unit
CG2       3.632628202    5.399640560    1.184555411 ILE  2
CT        C  -0.085
HG21      3.256929159    5.962747097    2.057613134 ILE  2
HC        H   0.029
HG22      4.728721142    5.525658131    1.229067683 ILE  2
HC        H   0.029
HG23      3.277012348    5.929985046    0.281316549 ILE  2
HC        H   0.029
! rigid unit 2
3 !atoms in this rigid unit
CG1       1.625806093    3.868085861    1.310235620 ILE  2
CT        C  -0.049
HG11      1.169472456    4.395492077    0.450418025 ILE  2
HC        H   0.027
HG12      1.273633957    2.823534966    1.211708426 ILE  2
HC        H   0.027
! rigid unit 3
4 !atoms in this rigid unit
CD1       1.028863907    4.391342163    2.632859945 ILE  2
CT        C  -0.085
HD11     -0.068560459    4.262083530    2.654643297 ILE  2
HC        H   0.028
HD12      1.436750174    3.852109432    3.508637428 ILE  2
HC        H   0.028
HD13      1.222232699    5.468014240    2.787941933 ILE  2
HC        H   0.028
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1 !ending of incoming bond and nn
```

```
3.783586502    3.069634438    -0.000003350  !beginning of bond
2 !bonds out
1 !unit bond d to
0 1 -1 -1 -1    ! beginning of outgoing bond and nn
3.632628202    5.399640560    1.184555411   !ending of outgoing
bond
2 !unit bonded to
0 1 -1 -1 -1    ! beginning of outgoing bond and nn
 1.625806093    3.868085861    1.310235620   !ending of outgoing
bond
! rigid unit 1
0 1 2 3 -1 !ending of incoming bond and nn
 3.184130907    3.905461311    1.203313947 !beginning of incoming
bond
0! bonds out
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond and nn
 3.184130907    3.905461311    1.203313947 !beginning of incoming
bond
1 !bonds out
3 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
1.028863907    4.391342163    2.632859945   !ending of outgoing
bond
! rigid unit 3
0 1 2 3 -1 !ending of incoming bond and nn
1.625806093    3.868085861    1.310235620 !beginning of bond
0 !bonds out
```

****

DATA FILE FOR LYSINE - K.DAT

****

```
! The side-chain structure file for Lysine
5 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.218223095    3.829745770    1.231236458 LYS  2
```

```
CT       C  -0.098
HB1       2.112416506    3.764609814    1.234413505 LYS  2
HC       H   0.038
HB2       3.536234617    3.317805290    2.163102627 LYS  2
HC       H   0.038
! rigid unit 1
3 !atoms in this rigid unit
CG        3.638167858    5.320005417    1.281187057 LYS  2
CT       C  -0.160
HG1       4.741127968    5.406830788    1.274424553 LYS  2
HC       H   0.116
HG2       3.295989990    5.833013058    0.360635072 LYS  2
HC       H   0.116
! rigid unit 2
3 !atoms in this rigid unit
CD        3.153400660    6.084614754    2.516160011 LYS  2
CT       C  -0.180
HD1       2.046517849    6.074027538    2.552636147 LYS  2
HC       H   0.122
HD2       3.501233101    5.571547031    3.435809374 LYS  2
HC       H   0.122
! rigid unit 3
3 !atoms in this rigid unit
CE        3.699187756    7.518018246    2.469964743 LYS  2
CT       C  -0.038
HE1       4.805956841    7.515174866    2.558616400 LYS  2
HC       H   0.098
HE2       3.475801945    8.000639915    1.495867610 LYS  2
HC       H   0.098
! rigid unit 4
4 !atoms in this rigid unit
NZ        3.098134756    8.306216240    3.560437918 LYS  2
N3       N  -0.138
HZ1       3.463554621    9.268757820    3.530759573 LYS  2
H3       H   0.294
HZ2       2.074491024    8.324481964    3.447653770 LYS  2
H3       H   0.294
HZ3       3.335658073    7.877095222    4.466163158 LYS  2
H3       H   0.294
```

```
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586502    3.069634914    -0.000003353 !beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
3.638167858     5.320005417     1.281187057   !ending of outgoing
bond
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond and nn
 3.218223095    3.829745770    1.231236458!beginning of bond
1 !bonds out
2 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
3.153400660     6.084614754     2.516160011 !ending of outgoing
bond
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond and nn
3.638167858    5.320005417    1.281187057 !beginning of bond
1 !bonds out
3 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
3.699187756     7.518018246     2.469964743 !ending of outgoing
bond
! rigid unit 3
0 1 2 -1 -1 !ending of incoming bond and nn
3.153400660    6.084614754    2.516160011!beginning of bond
1 !bonds out
4 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
 3.098134756    8.306216240    3.560437918 !ending of outgoing
bond
! rigid unit 4
0 1 2 3 -1 !ending of incoming bond and nn
3.699187756    7.518018246    2.469964743!beginning of bond
0 !bonds out

****************************************************************
```

DATA FILE FOR LEUCINE - L.DAT

```
*****************************************************************

! The side-chain structure file for Leucine
4 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB         3.217977524    3.860693455    1.213688374 LEU  2
CT        C  -0.061
HB1        3.617908239    3.413237095    2.146348953 LEU  2
HC        H   0.033
HB2        3.641148329    4.884153843    1.193638206 LEU  2
HC        H   0.033
! rigid unit 1
2 !atoms in this rigid unit
CG         1.676206470    3.974944353    1.357627273 LEU  2
CT        C  -0.010
HG         1.273801684    2.962582827    1.570222020 LEU  2
HC        H   0.031
! rigid unit 2
4 !atoms in this rigid unit
CD1        1.322771311    4.880306721    2.545703411 LEU  2
CT        C  -0.107
HD11       0.229164675    4.936426640    2.704123735 LEU  2
HC        H   0.034
HD12       1.758654118    4.507015228    3.491832256 LEU  2
HC        H   0.034
HD13       1.684926391    5.916738033    2.406197309 LEU  2
HC        H   0.034
! rigid unit 3
4 !atoms in this rigid unit
CD2        0.998154640    4.504262924    0.083184890 LEU  2
CT        C  -0.107
HD21      -0.093163513    4.622812748    0.214309067 LEU  2
HC        H   0.034
HD22       1.406615853    5.481475830   -0.234147355 LEU  2
HC        H   0.034
HD23       1.130140185    3.802904606   -0.761629283 LEU  2
```

```
HC       H   0.034
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
3.783586502    3.069634438    -0.000003367!beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1    ! beginning of outgoing bond and nn
1.676206470    3.974944353    1.357627273   !ending of outgoing
bond
! rigid unit 1
0 1 -1 -1 -1 !ending of incoming bond and nn
 3.184130907   3.905461311   1.203313947 !beginning of incoming
bond
2! bonds out
2 !unit bonded to
0 1 -1 -1 -1    ! beginning of outgoing bond and nn
1.322771311    4.880306721    2.545703411   !ending of outgoing
bond
3 !unit bonded to
0 1 -1 -1 -1    ! beginning of outgoing bond and nn
0.998154640    4.504262924    0.083184890  !ending of outgoing
bond
! rigid unit 2
0 1 2 3 -1 !ending of incoming bond and nn
1.676206470    3.974944353    1.357627273 !beginning of incoming
bond
0 !bonds out
! rigid unit 3
0 1 2 3 -1 !ending of incoming bond and nn
1.676206470    3.974944353    1.357627273 !beginning of bond
0 !bonds out

************************************************************
               DATA FILE FOR METHIONINE - M.DAT
************************************************************

! The side-chain structure file for Methionine
4 !rigid units in side-chain
```

```
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB          3.219568014    3.840672970    1.225060582 MET  2
CT        C  -0.151
HB1         3.547865868    3.348565578    2.163037539 MET  2
HC        H   0.027
HB2         3.671003819    4.850576401    1.262409329 MET  2
HC        H   0.027
! rigid unit 1
3 !atoms in this rigid unit
CG          1.685955524    4.011272907    1.265707970 MET  2
CT        C  -0.054
HG1         1.291312337    4.382569790    0.302083224 MET  2
HC        H   0.0652
HG2         1.199923158    3.034499168    1.452733874 MET  2
HC        H   0.0652
! rigid unit 2
3 !atoms in this rigid unit
SD          1.234688163    5.162067413    2.574714422 MET  2
S         S   0.737
LD1         1.486726403    6.202064514    2.319993973 MET  2
LP        L  -0.381
LD2         1.747960329    4.937880516    3.521441460 MET  2
LP        L  -0.381
! rigid unit 3
4 !atoms in this rigid unit
CE         -0.532971203    4.837210655    2.617241383 MET  2
CT        C  -0.134
HE1        -0.987815082    4.991072178    1.622043610 MET  2
HC        H   0.0652
HE2        -1.033426285    5.510134220    3.335405111 MET  2
HC        H   0.0652
HE3        -0.725545764    3.794905424    2.929581165 MET  2
HC        H   0.0652
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 ! nding of incoming bond and nn
3.783586502    3.069634438    -0.000003354 !beginning of bond
```

```
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
1.685955524     4.011272907     1.265707970 !ending of outgoing
bond
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond and nn
3.219568014    3.840672970    1.225060582 !beginning of bond
1 !bonds out
2 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
1.234688163     5.162067413     2.574714422 !ending of outgoing
bond
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond and nn
1.685955524    4.011272907    1.265707970 !beginning of bond
1 !bonds out
3 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 -0.532971203    4.837210655    2.617241383 !ending of outgoing
bond
! rigid unit 3
0 1 2 3 -1 !ending of incoming bond and nn
1.234688163    5.162067413    2.574714422!beginning of bond
0 !bonds out

******************************************************************
                 DATA FILE FOR APSARAGINE - N.DAT
******************************************************************

! The side-chain structure file for Asparagine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.222899199    3.830397844    1.236912012 ASN  2
CT       C  -0.086
HB1       3.611397266    3.364436865    2.163546562 ASN  2
HC       H   0.038
```

```
HB2       3.616078854    4.863478184    1.264652491 ASN  2
HC        H   0.038
! rigid unit 1
5 !atoms in this rigid unit
CG        1.698638678    3.892561436    1.381467938 ASN  2
C         C   0.675
OD1       1.085211635    3.155725241    2.139311790 ASN  2
O         O  -0.470
ND2       1.031797171    4.746669292    0.652490914 ASN  2
N         N  -0.867
HD21      0.019928589    4.602556705    0.711063743 ASN  2
H         H   0.344
HD22      1.562326550    5.282481670   -0.034363598 ASN  2
H         H   0.344
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634438   -0.000003353 !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
1.698638678    3.892561436    1.381467938  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.222899199   3.830397844   1.236912012 !beginning of bond for
unit 1
0 !bonds out from rigid unit 1

*************************************************************
               DATA FILE FOR GLUTAMINE - Q.DAT
*************************************************************

! The side-chain structure file for Glutamine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
```

```
CB         3.221223593    3.805351734    1.236027122 GLN  2
CT        C  -0.098
HB1        2.115758896    3.733683825    1.223282218 GLN  2
HC        H   0.038
HB2        3.538368225    3.258102417    2.148239136 GLN  2
HC        H   0.038
! rigid unit 1
3 !atoms in this rigid unit
CG         3.619170427    5.311230183    1.384292126 GLN  2
CT        C  -0.102
HG1        4.719832420    5.417502403    1.395145655 GLN  2
HC        H   0.057
HG2        3.298108339    5.879051685    0.491232127 GLN  2
HC        H   0.057
! rigid unit 2
5 !atoms in this rigid unit
CD         3.148421526    6.090956688    2.618209839 GLN  2
C         C   0.675
OE1        3.471138716    7.255728722    2.789397001 GLN  2
O         O  -0.470
NE2        2.408394814    5.500250816    3.521779537 GLN  2
N         N  -0.867
HE21       2.231919527    4.508390427    3.353902817 GLN  2
H         H   0.344
HE22       2.192787886    6.069860935    4.342392445 GLN  2
H         H   0.344
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634438   -0.000003353  !beginning of bond for
unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.619170427    5.311230183    1.384292126 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 1 and nn
3.221223593    3.805351734    1.236027122 !beginning of bond for
```

```
unit 1
1 !bonds out from rigid unit 0
2 !unit 1 is bonded to unit 2
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.148421526    6.090956688    2.618209839  !ending of outgoing
bond for unit 2
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond for unit 2 and nn
 3.619170427   5.311230183   1.384292126 !beginning of bond for
unit 2
0 !bonds out from rigid unit 2

************************************************************
                DATA FILE FOR ARGININE - R.DAT
************************************************************

! The side-chain structure file for Arginine
4 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.207483053    3.819248199    1.232642174 ARG  2
CT       C  -0.080
HB1       2.121760130    3.616136551    1.319550753 ARG  2
HC       H   0.056
HB2       3.644849300    3.393733978    2.159598827 ARG  2
HC       H   0.056
! rigid unit 1
3 !atoms in this rigid unit
CG        3.412360668    5.357305527    1.216631651 ARG  2
CT       C  -0.103
HG1       4.487451553    5.614737511    1.132990837 ARG  2
HC       H   0.074
HG2       2.938670874    5.796108723    0.315252036 ARG  2
HC       H   0.074
! rigid unit 2
3 !atoms in this rigid unit
CD        2.850392818    6.038671017    2.471077681 ARG  2
CT       C  -0.228
```

```
HD1       1.769480824    5.816972256    2.580044270 ARG  2
HC       H   0.133
HD2       3.353989840    5.649005413    3.379585028 ARG  2
HC       H   0.133
! rigid unit 3
9 !atoms in this rigid unit
NE        3.069616079    7.502031326    2.345978022 ARG  2
N2       N  -0.324
HE        3.539865971    7.837357998    1.493146777 ARG  2
H3       H   0.269
CZ        2.710799694    8.413488388    3.240067959 ARG  2
CA       C   0.760
NH1       2.972572088    9.643490791    2.971310854 ARG  2
N2       N  -0.624
HH11      3.439955235    9.745957375    2.068439484 ARG  2
H3       H   0.361
HH12      2.697422743   10.348603249    3.651821136 ARG  2
H3       H   0.361
NH2       2.114365101    8.144207001    4.363539696 ARG  2
N2       N  -0.624
HH21      1.888047814    8.930854797    4.969158173 ARG  2
H3       H   0.361
HH22      1.947107434    7.146794796    4.499028206 ARG  2
H3       H   0.361
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.783586502   3.069634914   -0.000003315 !beginning of bond for
unit 0
1 !bond out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1 ! beginning of outgoing bond and nn
3.412360668    5.357305527    1.216631651 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
 3.207483053   3.819248199   1.232642174 !beginning of bond for
unit 1
1 !bond out from rigid unit 1
```

```
2 !unit 1 is bond d to unit 2
0 1 2 -1 -1 ! beginning of outgoing bond and nn
2.850392818    6.038671017    2.471077681 !ending of outgoing
bond
! rigid unit 2
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
3.412360668   5.357305527   1.216631651 !beginning of bond for
unit 2
1 !bond out from rigid unit 2
3 !unit 2 is bonded to unit 3
0 1 2 -1 -1 ! beginning of outgoing bond and nn
3.069616079    7.502031326    2.345978022 !ending of outgoing
bond
! rigid unit 3
0 1 2 -1 -1 !ending of incoming bond for unit 0 and nn
2.850392818   6.038671017   2.471077681!beginning of bond for
unit 3
0 !bonds out from rigid unit 3

*************************************************************
                DATA FILE FOR SERINE - S.DAT
*************************************************************

! The side-chain structure file for Serine
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.203660250    3.871555328    1.191825747 SER  2
CT       C   0.018
HB1       3.445731640    4.945727825    1.071671009 SER  2
HC       H   0.119
HB2       2.097403765    3.828571320    1.202566266 SER  2
HC       H   0.119
! rigid unit 1
2 !atoms in this rigid unit
OG        3.711599350    3.433972597    2.457015276 SER  2
OH       O  -0.550
HG        3.430009127    2.523327112    2.580434084 SER  2
```

```
HO        H    0.310
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 ! nding of incoming bond for unit 0 and nn
 3.783586502    3.069634438    -0.000003353  !beginning of bond
for unit 0
1 !bonds out from rigid unit 0
1 !unit 0 is bonded to unit 1
0 1 2 -1 -1   ! beginning of outgoing bond and nn
3.711599350     3.433972597     2.457015276 !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 -1 -1 -1 !ending of incoming bond for unit 1 and nn
 3.203660250    3.871555328    1.191825747  !beginning of bond
for unit 1
0 !bonds out from rigid unit 1

************************************************************
               DATA FILE FOR THREONINE - T.DA
************************************************************

! The side-chain structure file for Threonine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
2 !atoms in this rigid unit
CB         3.220216751    3.864162445    1.226425409 THR  2
CT        C   0.170
HB         3.504307270    3.322291374    2.154003382 THR  2
HC        H   0.082
! rigid unit 1
2 !atoms in this rigid unit
OG1        1.802008867    3.940322876    1.161503792 THR  2
OH        O  -0.550
HG1        1.520381451    4.374082565    1.972538352 THR  2
HO        H   0.310
! rigid unit 2
4 !atoms in this rigid unit
CG2        3.680637360    5.331728935    1.361316323 THR  2
```

```
CT        C  -0.191
HG21       3.224400043     5.832503796     2.234619141 THR  2
HC        H   0.065
HG22       4.774106026     5.420624733     1.502453089 THR  2
HC        H   0.065
HG23       3.418393373     5.928008556     0.466874599 THR  2
HC        H   0.065
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1 !ending of incoming bond and nn
3.783586502    3.069634438    -0.000003353 !beginning of bond
2 !bonds out
1 !unit 0 is bonded
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
1.802008867     3.940322876     1.161503792 !ending of outgoing
bond for unit 0
2 !unit 0 is bonded
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
3.680637360     5.331728935     1.361316323  !ending of outgoing
bond for unit 0
! rigid unit 1
0 1 -1 -1 -1 !ending of incoming bond and nn
 3.220216751     3.864162445     1.226425409  !beginning of bond
for unit 1
0 !bonds out
! rigid unit 2
0 1 2 3 -1 !ending of incoming bond and nn
 3.220216751     3.864162445     1.226425409  !beginning of bond
for unit 1
0 !bonds out

***************************************************************
                 DATA FILE FOR VALINE - V.DAT
***************************************************************

! The side-chain structure file for Valine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
```

```
2 !atoms in this rigid unit
CB        3.211601496    3.852613449    1.247815728 VAL  2
CT       C  -0.012
HB        3.447319269    3.248452187    2.150032282 VAL  2
HC       H   0.024
! rigid unit 1
4 !atoms in this rigid unit
CG1       1.676198244    4.045934200    1.217347741 VAL  2
CT       C  -0.091
HG11      1.351996183    4.697401524    0.384493083 VAL  2
HC       H   0.031
HG12      1.142809749    3.084587097    1.106773376 VAL  2
HC       H   0.031
HG13      1.300095797    4.498250008    2.155061245 VAL  2
HC       H   0.031
! rigid unit 2
4 !atoms in this rigid unit
CG2       3.797980547    5.269292355    1.500991821 VAL  2
CT       C  -0.091
HG21      3.634918213    5.953960419    0.647068620 VAL  2
HC       H   0.031
HG22      3.359194279    5.751780510    2.395626068 VAL  2
HC       H   0.031
HG23      4.886912346    5.247161865    1.696415067 VAL  2
HC       H   0.031
! BOND INFORMATION
! rigid unit 0
0 1 -1 -1 -1 !ending of incoming bond and nn
3.783586502   3.069634438   -0.000003354 !beginning of bond
2 !bonds out
1 !unit bonded to
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
 1.676198244    4.045934200    1.217347741!ending of outgoing
bond
2 !unit bonded to
0 1 -1 -1 -1   ! beginning of outgoing bond and nn
 3.797980547    5.269292355    1.500991821!ending of outgoing
bond
! rigid unit 1
```

```
0 1 2 3 -1 !ending of incoming bond and nn
3.211601496   3.852613449   1.247815728  !beginning of outgoing
bond
0 !bonds out
! rigid unit 2
0 1 2 3 -1 !ending of incoming bond and nn
3.211601496   3.852613449   1.247815728  !beginning of outgoing
bond
0 !bonds out
****************************************************************
               DATA FILE FOR TRYPTOPHAN - W.DAT
****************************************************************

! The side-chain structure file for Tryptophan
2 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB        3.247885227    3.809360981    1.256884575 TRP  2
CT       C  -0.098
HB1       3.555066347    3.270197153    2.175767183 TRP  2
HC       H   0.038
HB2       3.728011608    4.802421093    1.350249052 TRP  2
HC       H   0.038
! rigid unit 1
15 !atoms in this rigid unit
CG        1.731538415    4.025276661    1.276940465 TRP  2
C*       C  -0.135
CD1       0.792832434    3.205200195    1.936712861 TRP  2
CW       C   0.044
NE1      -0.527979255    3.628766537    1.692452073 TRP  2
NA       N  -0.352
CE2      -0.376119167    4.727549076    0.861387193 TRP  2
CN       C   0.154
CD2       0.994750261    4.975831032    0.602216363 TRP  2
CB       C   0.146
HD1       1.058894038    2.330861330    2.516448259 TRP  2
HC       H   0.093
HE1      -1.402328849    3.197247982    2.011827707 TRP  2
```

```
H        H   0.271
CE3        1.387488961    6.039774895   -0.250452638 TRP  2
CA       C  -0.173
HE3        2.430646658    6.226261139   -0.463923573 TRP  2
HC       H   0.086
CZ3        0.392907262    6.841813087   -0.810243368 TRP  2
CA       C  -0.066
HZ3        0.674497783    7.661212444   -1.455789328 TRP  2
HC       H   0.057
CH2       -0.963685811    6.602497578   -0.548699141 TRP  2
CA       C  -0.077
HH2       -1.710847259    7.243553162   -0.992942095 TRP  2
HC       H   0.074
CZ2       -1.364877820    5.549452305    0.277642310 TRP  2
CA       C  -0.168
HZ2       -2.410887718    5.363564491    0.470484644 TRP  2
HC       H   0.084
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
 3.783586740    3.069634914   -0.000003497 !beginning of bond
1 !bonds out
1 !unit 0 is bonded
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 1.731538415    4.025276661    1.276940465!ending of outgoing
bond for unit 0
! rigid unit 1
0 1 4 -1 -1 !ending of incoming bond and nn
 3.247885227    3.809360981    1.256884575  !beginning of bond
for unit 1
0 !bonds out
```

```
*************************************************************
             DATA FILE FOR TYROSINE - Y.DAT
*************************************************************

! The side-chain structure fil  for Tyrosine
3 !rigid units in side-chain
! ATOM INFORMATION
! rigid unit 0
3 !atoms in this rigid unit
CB          3.293353796     3.842515945     1.259159327 TYR  2
CT        C  -0.098
HB1         3.703839302     3.358918667     2.169649363 TYR  2
HC        H   0.038
HB2         3.749134064     4.852351665     1.277104497 TYR  2
HC        H   0.038
! rigid unit 1
10 !atoms in this rigid unit
CG          1.778211594     4.019127369     1.411828637 TYR  2
CA        C  -0.030
CD1         1.068759203     3.196300983     2.292453527 TYR  2
CA        C  -0.002
HD1         1.585003138     2.435774803     2.862824917 TYR  2
HC        H   0.064
CD2         1.095163584     4.989490032     0.672801077 TYR  2
CA        C  -0.002
HD2         1.629922271     5.630218983    -0.014210327 TYR  2
HC        H   0.064
CE1        -0.309100747     3.338460445     2.427857637 TYR  2
CA        C  -0.264
HE1        -0.845880806     2.691843510     3.105883360 TYR  2
HC        H   0.102
CZ         -0.983952701     4.304777145     1.686211467 TYR  2
C         C   0.462
CE2        -0.283983082     5.129064560     0.809688389 TYR  2
CA        C  -0.264
HE2        -0.814125061     5.873366833     0.234044328 TYR  2
HC        H   0.102
! rigid unit 1
2 !atoms in this rigid unit
OH         -2.337103367     4.443373203     1.815491915 TYR  2
OH        O  -0.528
HH         -2.648404837     3.798558235     2.453088284 TYR  2
HO        H   0.334
! BOND INFORMATION
! rigid unit 0
0 1 2 -1 -1 !ending of incoming bond and nn
```

```
3.783586264    3.069634914   -0.000003354 !beginning of bond
1 !bonds out
1 !unit bonded to
0 1 2 -1 -1   ! beginning of outgoing bond and nn
 1.778211594    4.019127369     1.411828637!ending of outgoing
bond for unit 0
! rigid unit 1
0 1 3 -1 -1 !ending of incoming bond and nn
 3.293353796    3.842515945    1.259159327  !beginning of bond
for unit 1
1 !bonds out
2 !unit bonded to
7 5 8 -1 -1   ! beginning of outgoing bond and nn
-2.337103367    4.443373203    1.815491915 !ending of outgoing
bond for unit 0
! rigid unit 2
0 1 -1 -1 -1 !ending of incoming bond and nn
-0.983952701    4.304777145    1.686211467  !beginning of bond
for unit 1
0 !bonds out
```

```
************************************************************
           DATA FILE FOR INITIAL PROTOTYPE - CX6C.CAR
************************************************************
```

```
!BIOSYM archive 3
PBC=OFF
!DATE Thu Mar  2 10:02:29 1995
SG          0.051616628    8.775964550    2.653307337 CYSn 1
S        S   0.824
LG1        -0.116704460    8.906803991    3.732450018 CYSn 1
LP       L  -0.405
LG2        -0.816371929    8.216369655    2.274560255 CYSn 1
LP       L  -0.405
CB          1.625257994    7.970290997    2.280061368 CYSn 1
CT       C  -0.098
HB1         1.743097230    7.117856362    2.972980432 CYSn 1
HC       H   0.050
HB2         2.457560406    8.667686711    2.506611212 CYSn 1
```

```
HC      H   0.050
CA       1.664891168     7.503978115     0.811322158 CYSn 1
CT      C   0.035
HA       2.715618613     7.453348875     0.469159517 CYSn 1
HC      H   0.032
N        0.954382540     8.512673633     0.003030230 CYSn 1
NT      N  -0.463
C        1.063568189     6.132700222     0.616111991 CYSn 1
C       C   0.616
O        0.248707622     5.654726837     1.414398016 CYSn 1
O       O  -0.504
N        1.449902196     5.479885680    -0.464156147 GLY  2
N       N  -0.463
HN       2.157106102     5.992384244    -1.099457509 GLY  2
H       H   0.252
CA       0.868490592     4.154014497    -0.652902307 GLY  2
CT      C   0.035
HA1      1.550908149     3.403064022    -0.212395307 GLY  2
HC      H   0.032
HA2     -0.097660558     4.132736815    -0.116611463 GLY  2
HC      H   0.032
C        0.730531165     3.827591429    -2.120728786 GLY  2
C       C   0.616
O        1.559375145     4.206208097    -2.957020570 GLY  2
O       O  -0.504
N       -0.320742949     3.103195380    -2.456098946 GLY  3
N       N  -0.463
HN      -0.976177839     2.817016114    -1.646836012 GLY  3
H       H   0.252
CA      -0.454134161     2.787581074    -3.875321662 GLY  3
CT      C   0.035
HA1     -0.907422830     1.783240810    -3.972773051 GLY  3
HC      H   0.032
HA2     -1.127648566     3.540414569    -4.323795441 GLY  3
HC      H   0.032
C        0.896974016     2.736484179    -4.547627543 GLY  3
C       C   0.616
O        1.315189212     1.712629073    -5.101282348 GLY  3
O       O  -0.504
```

```
N          1.599575272    3.853622667   -4.520184621 GLY  4
N         N  -0.463
HN         1.137216234    4.691535216   -4.019658253 GLY  4
H         H   0.252
CA         2.905944550    3.804217731   -5.170228610 GLY  4
CT        C   0.035
HA1        3.056204584    2.789614618   -5.584558431 GLY  4
HC        H   0.032
HA2        2.897891721    4.540755026   -5.994216851 GLY  4
HC        H   0.032
C          4.014980067    4.050747291   -4.175561433 GLY  4
C         C   0.616
O          4.978871195    4.780583329   -4.436272241 GLY  4
O         O  -0.504
N          3.887759074    3.450944950   -3.006608050 GLY  5
N         N  -0.463
HN         3.003276191    2.844372268   -2.879487738 GLY  5
H         H   0.252
CA         4.960071382    3.689311240   -2.044877031 GLY  5
CT        C   0.035
HA1        5.709592998    2.881830301   -2.144167698 GLY  5
HC        H   0.032
HA2        5.427393718    4.658369322   -2.297948016 GLY  5
HC        H   0.032
C          4.437174470    3.643619035   -0.629041435 GLY  5
C         C   0.616
O          3.798322352    2.676595378   -0.197242766 GLY  5
O         O  -0.504
N          4.713663113    4.691871185    0.124033264 GLY  6
N         N  -0.463
HN         5.286002166    5.476492875   -0.348403798 GLY  6
H         H   0.252
CA         4.208080753    4.647691975    1.492986659 GLY  6
CT        C   0.035
HA1        3.303800182    4.010943092    1.515218779 GLY  6
HC        H   0.032
HA2        4.993057374    4.194323221    2.125265975 GLY  6
HC        H   0.032
C          3.799265981    6.023038258    1.963510280 GLY  6
```

```
C          C   0.616
O           4.006824522    7.036283245    1.285298717 GLY  6
O          O  -0.504
N           3.195690211    6.077750863    3.136158080 GLY  7
N          N  -0.463
HN          3.055107813    5.133307510    3.640799839 GLY  7
H          H   0.252
CA          2.800412417    7.407555656    3.591101372 GLY  7
CT         C   0.035
HA1         1.946687677    7.303619509    4.286815466 GLY  7
HC         H   0.032
HA2         3.660862081    7.847316876    4.127520148 GLY  7
HC         H   0.032
C           2.334578164    8.258959996    2.434291753 GLY  7
C          C   0.616
O           2.337411236    9.494643783    2.487154063 GLY  7
O          O  -0.504
N           1.936206121    7.605756209    1.358640986 CYSN 8
N          N  -0.463
HN          1.983632457    6.528240768    1.414418956 CYSN 8
H          H   0.252
CA          1.485796919    8.428968216    0.240136508 CYSN 8
CT         C   0.035
HA          0.399931102    8.271042216    0.100059529 CYSN 8
HC         H   0.032
C           2.167493478    8.018162291   -1.043072620 CYSN 8
C          C   0.616
CB          1.746659419    9.902481747    0.610166221 CYSN 8
CT         C  -0.098
HB1         2.709270705   10.016688002    1.140264476 CYSN 8
HC         H   0.050
HB2         1.816139488   10.541353385   -0.293951287 CYSN 8
HC         H   0.050
SG          0.440719361   10.532225816    1.688457720 CYSN 8
S          S   0.824
LG1        -0.404239097   10.957145937    1.126774557 CYSN 8
LP         L  -0.405
LG2         0.793091788   11.329491558    2.359427872 CYSN 8
LP         L  -0.405
```

```
end
end

*********** ************************************************
************************************************************
                        END OF LISTING
***************************************************************
***************************************************************

***************************************************************
************************************************************
             DATA FILE WEINER FORCES - AMBER.FRC
***************************************************************
**********************************************************

!BIOSYM forcefield          2
#version amber.frc  1.0  19-Oct-90
#version amber.frc  1.1   8-Aug-92
#define amber
> This is the new format version of the amber forcefield
!Ver  Ref              Function       Label
!---- ---   --------------------------------     ------
 1.0   1    atom_types                     amber
 1.0   1    equivalence                    amber
 1.0   1    hbond_definition               amber
 1.0   1    quadratic_bond                 amber
 1.0   1    quadratic_angle                amber
 1.0   1    torsion_3                      amber
 1.0   1    out_of_plane                   amber
 1.0   1    nonbond(12-6)                  amber
 1.0   1    hydrogen_bond(10-12)           amber
#atom_types    amber
> Atom type definitions for any variant of amber
> Masses from CRC 1973/74 pages B-250.
!Ver  Ref  Type    Mass       Element  Comment
!---- ---  ----  ----------  -------
 1.0   1    C     12.000000    C       Kollman's Field: Masses
from CRC 1973/74 pages B-250.
 1.0   1    C*    12.000000    C
```

1.0	1	C2	12.000000	C
1.0	3	C3	15.000000	C
1.0	1	CA	12.000000	C
1.0	1	CB	12.000000	C
1.0	1	CC	12.000000	C
1.0	3	CD	13.000000	C
1.0	3	CE	13.000000	C
1.0	3	CF	13.000000	C
1.0	3	CG	13.000000	C
1.0	3	CH	13.000000	C
1.0	3	CI	13.000000	C
1.0	3	CJ	13.000000	C
1.0	1	CK	12.000000	C
1.0	1	CM	12.000000	C
1.0	1	CN	12.000000	C
1.0	3	CP	13.000000	C
1.0	1	CQ	12.000000	C
1.0	1	CR	12.000000	C
1.0	1	CT	12.000000	C
1.0	1	CV	12.000000	C
1.0	1	CW	12.000000	C
1.0	1	H	1.007825	H
1.0	1	H2	1.007825	H
1.0	1	H3	1.007825	H
1.0	1	HC	1.007825	H
1.0	1	HO	1.007825	H
1.0	1	HS	1.007825	H
1.0	3	LP	3.000000	H
1.0	1	N	14.003070	N
1.0	1	N*	14.003070	N
1.0	1	N2	14.003070	N
1.0	1	N3	14.003070	N
1.0	1	NA	14.003070	N
1.0	1	NB	14.003070	N
1.0	1	NC	14.003070	N
1.0	1	NP	14.003070	N
1.0	1	NT	14.003070	N
1.0	1	O	15.994910	O
1.0	1	O2	15.994910	O

```
 1.0    1    OH    15.994910    O
 1.0    1    OS    15.994910    O
 1.0    1    P     30.993760    P
 1.0    1    S     31.972070    S
 1.0    1    SH    31.972070    S
 1.0    3    CO    40.080000    Ca
 1.0    3    HW    1.008000     H
 1.0    3    IM    35.450000    Cl
 1.0    3    CU    63.550000    Cu
 1.0    3    I     22.990000    I
 1.0    3    MG    24.305000    Mg
 1.0    3    OW    16.000000    O
 1.0    3    QC    132.90000    Cs
 1.0    3    QK    39.100000    K
 1.0    3    QL    6.940000     Li
 1.0    3    QN    22.990000    Na
 1.0    3    QR    85.470000    Rb
 1.1    4    CS    12.000000    C      carbohydrate sp3 carbon
 1.1    4   AC    12.000000    C      carbohydrate alpha-anomeric
carbon
 1.1    4    BC    12.000000    C      carbohydrate beta-anomeric
carbon
 1.1    4    HT     1.007825    H      carbohydrate sp3 hydro
 1.1    4   AH    1.007825    H      carbohydrate alpha-anomeric
hydrogen
 1.1    4    BH    1.007825    H      carbohydrate beta-anomeric
hydrogen
 1.1     4     HY      1.007825     H       carbohydrate hydroxyl
hydrogen
 1.1     4     OT     15.994910     O       carbohydrate hydroxyl
oxygen
 1.1    4    OA    15.994910    O      carbohydrate alpha-anomeric
oxygen
 1.1    4    OB    15.994910    O      carbohydrate beta-anomeric
oxygen
 1.1    4    OE    15.994910    O      carbohydrate ring oxygen
 1.0    1    h$    1.007825    H      Hydrogen atom for aTOMATIC
PARAMETER assignment
 1.0    1    c$    12.000000    C      Carbon atom for automatic
```

```
parameter assignment
 1.0    1    n$    14.003070    N      Nitrogen atom for automatic
parameter assignment
 1.0    1    o$    15.994910    O      Oxyg n atom for automatic
parameter assignment
 1.0    1    s$    31.972070    S      Sulfur atom for automatic
parameter assignment
 1.0    1    p$    30.993760    P      Phosphorous atom for
automatic parameter assignment
#equivalence   amber
> Equivalence table for any variant of amber
!                          Equivalences
!                 -----------------------------------------
!Ver  Ref   Type  NonB    Bond    Angle   Torsion  OOP
!---- ---   ----  ----    ----    -----   -------  ----
 1.0   1    C     C       C       C       C        C
 1.0   1    C*    C*      C*      C*      C*       C*
 1.0   1    C2    C2      C2      C2      C2       C2
 1.0   1    C3    C3      C3      C3      C3       C3
 1.0   1    CA    CA      CA      CA      CA       CA
 1.0   1    CB    CB      CB      CB      CB       CB
 1.0   1    CC    CC      CC      CC      CC       CC
 1.0   1    CD    CD      CD      CD      CD       CD
 1.0   1    CE    CE      CE      CE      CE       CE
 1.0   1    CF    CF      CF      CF      CF       CF
 1.0   1    CG    CG      CG      CG      CG       CG
 1.0   1    CH    CH      CH      CH      CH       CH
 1.0   1    CI    CI      CI      CI      CI       CI
 1.0   1    CJ    CJ      CJ      CJ      CJ       CJ
 1.0   1    CK    CK      CK      CK      CK       CK
 1.0   1    CM    CM      CM      CM      CM       CM
 1.0   1    CN    CN      CN      CN      CN       CN
 1.0   1    CP    CP      CP      CP      CP       CP
 1.0   1    CQ    CQ      CQ      CQ      CQ       CQ
 1.0   1    CR    CR      CR      CR      CR       CR
 1.0   1    CT    CT      CT      CT      CT       CT
 1.0   1    CV    CV      CV      CV      CV       CV
 1.0   1    CW    CW      CW      CW      CW       CW
 1.0   1    H     H       H       H       H        H
```

1.0	1	H2	H2	H2	H2	H2	H2
1.0	1	H3	H3	H3	H3	H3	H3
1.0	1	HC	HC	HC	HC	HC	HC
1.0	1	HO	HO	HO	HO	HO	HO
1.0	1	HS	HS	HS	HS	HS	HS
1.0	1	LP	LP	LP	LP	LP	LP
1.0	1	N	N	N	N	N	N
1.0	1	N*	N*	N*	N*	N*	N*
1.0	1	N2	N2	N2	N2	N2	N2
1.0	1	N3	N3	N3	N3	N3	N3
1.0	1	NA	NA	NA	NA	NA	NA
1.0	1	NB	NB	NB	NB	NB	NB
1.0	1	NC	NC	NC	NC	NC	NC
1.0	1	NP	NP	NP	NP	NP	NP
1.0	1	NT	NT	NT	NT	NT	NT
1.0	1	O	O	O	O	O	O
1.0	1	O2	O2	O2	O2	O2	O2
1.0	1	OH	OH	OH	OH	OH	OH
1.0	1	OS	OS	OS	OS	OS	OS
1.0	1	P	P	P	P	P	P
1.0	1	S	S	S	S	S	S
1.0	1	SH	SH	SH	SH	SH	SH
1.0	3	I	I	I	I	I	I
1.0	3	CU	CU	CU	CU	CU	CU
1.0	3	IM	IM	IM	IM	IM	IM
1.0	3	CO	CO	CO	CO	CO	CO
1.0	3	HW	HW	HW	HW	HW	HW
1.0	3	MG	MG	MG	MG	MG	MG
1.0	3	OW	OW	OW	OW	OW	OW
1.0	3	QC	QC	QC	QC	QC	QC
1.0	3	QK	QK	QK	QK	QK	QK
1.0	3	QL	QL	QL	QL	QL	QL
1.0	3	QN	QN	QN	QN	QN	QN
1.0	3	QR	QR	QR	QR	QR	QR
1.1	4	CS	CS	CS	CS	CS	CS
1.1	4	AC	AC	AC	AC	AC	AC
1.1	4	BC	BC	BC	BC	BC	BC
1.1	4	HT	HT	HT	HT	HT	HT
1.1	4	AH	AH	AH	AH	AH	AH

```
 1.1    4    BH    BH       BH       BH       BH        BH
 1.1    4    HY    HY       HY       HY       HY        HY
 1.1    4    OT    OT       OT       OT       OT        OT
 1.1    4    OA    OA       OA       OA       OA        OA
 1.1    4    OB    OB       OB       OB       OB        OB
 1.1    4    OE    OE       OE       OE       OE        OE
 1.0    1    h$    h$       h$       h$       h$        h$
 1.0    1    c$    c$       c$       c$       c$        c$
 1.0    1    n$    n$       n$       n$       n$        n$
 1.0    1    o$    o$       o$       o$       o$        o$
 1.0    1    s$    s$       s$       s$       s$        s$
 1.0    1    p$    p$       p$       p$       p$        p$
#hbond_definition   amber
 1.0    1   distance      2.5000
 1.0    1   angle        90.0000
 1.0    1   donors        H    HO   H2   H3   HS
 1.0    1   acceptors     NB   NC   O2   O    OH   S    SH
#quadratic_bond     amber
> E = K2 * (R - R0)^2
!Ver  Ref     I     J          R0          K2
!---- ---    ----  ----     -------    --------
 1.0   3     OW    HW       0.9572     553.0000
 1.0   3     HW    HW       1.5136     553.0000
 1.0   3     CH    N3       1.471      367.0000
 1.0   3     C3    SH       1.810      222.0000
 1.0   1     C     C2       1.5220     317.0000
 1.0   1     C     C3       1.5220     317.0000
 1.0   1     C     CA       1.4000     469.0000
 1.0   1     C     CB       1.4190     447.0000
 1.0   1     C     CD       1.4000     469.0000
 1.0   1     C     CH       1.5220     317.0000
 1.0   1     C     CJ       1.4440     410.0000
 1.0   1     C     CM       1.4440     410.0000
 1.0   3     C     CT       1.5220     317.0000
 1.0   1     C     N        1.3350     490.0000
 1.0   1     C     N*       1.3830     424.0000
 1.0   1     C     NA       1.3880     418.0000
 1.0   1     C     NC       1.3580     457.0000
 1.0   1     C     O        1.2290     570.0000
```

1.0	1	C	O2	1.2500	656.0000
1.0	1	C	OH	1.3640	450.0000
1.0	1	C*	C2	1.4950	317.0000
1.0	1	C*	CB	1.4590	388.0000
1.0	1	C*	CG	1.3520	546.0000
1.0	1	C*	CT	1.4950	317.0000
1.0	1	C*	CW	1.3520	546.0000
1.0	1	C*	HC	1.0800	340.0000
1.0	1	C2	C2	1.5260	260.0000
1.0	1	C2	C3	1.5260	260.0000
1.0	1	C2	CA	1.5100	317.0000
1.0	1	C2	CC	1.5040	317.0000
1.0	1	C2	CH	1.5260	260.0000
1.0	1	C2	N	1.4490	337.0000
1.0	1	C2	N2	1.4630	337.0000
1.0	1	C2	N3	1.4710	367.0000
1.0	1	C2	NT	1.4710	367.0000
1.0	1	C2	OH	1.4250	386.0000
1.0	1	C2	OS	1.4250	320.0000
1.0	1	C2	S	1.8100	222.0000
1.0	1	C2	SH	1.8100	222.0000
1.0	1	C3	CH	1.5260	260.0000
1.0	1	C3	CM	1.5100	317.0000
1.0	1	C3	N	1.4490	337.0000
1.0	1	C3	N*	1.4750	337.0000
1.0	1	C3	N2	1.4630	337.0000
1.0	1	C3	N3	1.4710	367.0000
1.0	1	C3	OH	1.4250	386.0000
1.0	1	C3	OS	1.4250	320.0000
1.0	1	C3	S	1.8100	222.0000
1.0	1	CA	CA	1.4000	469.0000
1.0	1	CA	CB	1.4040	469.0000
1.0	1	CA	CD	1.4000	469.0000
1.0	1	CA	CJ	1.4330	427.0000
1.0	1	CA	CM	1.4330	427.0000
1.0	1	CA	CN	1.4000	469.0000
1.0	1	CA	CT	1.5100	317.0000
1.0	1	CA	HC	1.0800	340.0000
1.0	1	CA	N2	1.3400	481.0000

1.0	1	CA	NA	1.3810	427.0000
1.0	1	CA	NC	1.3390	483.0000
1.0	1	CB	CB	1.3700	520.0000
1.0	1	CB	CD	1.4000	469.0000
1.0	1	CB	CN	1.4190	447.0000
1.0	1	CB	N*	1.3740	436.0000
1.0	1	CB	NB	1.3910	414.0000
1.0	1	CB	NC	1.3540	461.0000
1.0	1	CC	CF	1.3750	512.0000
1.0	1	CC	CG	1.3710	518.0000
1.0	1	CC	CT	1.5040	317.0000
1.0	1	CC	CV	1.3750	512.0000
1.0	1	CC	CW	1.3710	518.0000
1.0	1	CC	NA	1.3850	422.0000
1.0	1	CC	NB	1.3940	410.0000
1.0	1	CD	CD	1.4000	469.0000
1.0	1	CD	CN	1.4000	469.0000
1.0	1	CE	N*	1.3710	440.0000
1.0	1	CE	NB	1.3040	529.0000
1.0	1	CF	NB	1.3940	410.0000
1.0	1	CG	NA	1.3810	427.0000
1.0	1	CH	CH	1.5260	260.0000
1.0	1	CH	N	1.4490	337.0000
1.0	1	CH	N*	1.4750	337.0000
1.0	1	CH	NT	1.4710	367.0000
1.0	1	CH	OH	1.4250	386.0000
1.0	1	CH	OS	1.4250	320.0000
1.0	1	CI	NC	1.3240	502.0000
1.0	1	CJ	CJ	1.3500	549.0000
1.0	1	CJ	CM	1.3500	549.0000
1.0	1	CJ	N*	1.3650	448.0000
1.0	1	CK	HC	1.0800	340.0000
1.0	1	CK	N*	1.3710	440.0000
1.0	1	CK	NB	1.3040	529.0000
1.0	1	CM	CM	1.3500	549.0000
1.0	1	CM	CT	1.5100	317.0000
1.0	1	CM	HC	1.0800	340.0000
1.0	1	CM	N*	1.3650	448.0000
1.0	1	CN	NA	1.3800	428.0000

1.0	1	CP	NA	1.3430	477.0000
1.0	1	CP	NB	1.3350	488.0000
1.0	1	CQ	HC	1.0800	340.0000
1.0	1	CQ	NC	1.3240	502.0000
1.0	1	CR	HC	1.0800	340.0000
1.0	1	CR	NA	1.3430	477.0000
1.0	1	CR	NB	1.3350	488.0000
1.0	1	CT	CT	1.5260	310.0000
1.0	1	CT	HC	1.0900	331.0000
1.0	3	CT	N	1.4490	337.0000
1.0	1	CT	N*	1.4750	337.0000
1.0	1	CT	N2	1.4630	337.0000
1.0	1	CT	N3	1.4710	367.0000
1.0	1	CT	OH	1.4100	320.0000
1.0	1	CT	OS	1.4100	320.0000
1.0	1	CT	S	1.8100	222.0000
1.0	1	CT	SH	1.8100	222.0000
1.0	1	CV	HC	1.0800	340.0000
1.0	1	CV	NB	1.3940	410.0000
1.0	1	CW	HC	1.0800	340.0000
1.0	1	CW	NA	1.3810	427.0000
1.0	1	H	N	1.0100	434.0000
1.0	1	H	N2	1.0100	434.0000
1.0	1	H	NA	1.0100	434.0000
1.0	1	H	N*	1.0100	434.0000
1.0	1	H2	N	1.0100	434.0000
1.0	1	H2	N2	1.0100	434.0000
1.0	1	H2	NT	1.0100	434.0000
1.0	1	H3	N2	1.0100	434.0000
1.0	1	H3	N3	1.0100	434.0000
1.0	1	HO	OH	0.9600	553.0000
1.0	1	HO	OS	0.9600	553.0000
1.0	1	HS	SH	1.3360	274.0000
1.0	3	LP	S	0.6790	150.0000
1.0	3	LP	SH	0.6790	150.0000
1.0	1	O2	P	1.4800	525.0000
1.0	1	OH	P	1.6100	230.0000
1.0	1	OS	P	1.6100	230.0000
1.0	1	S	S	2.0380	166.0000

1.1	4	OH	HO	0.9600	553.0000
1.1	4	OT	HY	0.9720	460.5000
1.1	4	OA	HY	0.9720	460.5000
1.1	4	OB	HY	0.9720	460.5000
1.1	4	CS	HT	1.0990	337.3000
1.1	4	AC	AH	1.0990	337.3000
1.1	4	BC	BH	1.0990	337.3000
1.1	4	AC	HT	1.0990	337.3000
1.1	4	BC	HT	1.0990	337.3000
1.1	4	AC	OA	1.4110	334.3000
1.1	4	BC	OB	1.3900	334.3000
1.1	4	CS	OA	1.4400	334.3000
1.1	4	CS	OB	1.4400	334.3000
1.1	4	CS	CS	1.5230	214.8000
1.1	4	CS	CT	1.5230	214.8000
1.1	4	AC	CS	1.5230	214.8000
1.1	4	BC	CS	1.5230	214.8000
1.1	4	CS	OT	1.4110	334.3000
1.1	4	CS	OE	1.4270	296.7000
1.1	4	AC	OE	1.4270	296.7000
1.1	4	BC	OE	1.4270	296.7000
1.1	4	CS	N	1.4490	355.0000
1.1	4	H	N	1.0100	434.0000
1.1	4	C	N	1.3350	490.0000
1.1	4	C	O	1.2290	570.0000
1.1	4	C	CS	1.5220	335.0000
1.0	1	C$1	C$1	1.5260	260.0000
1.0	1	C$2	C$2	1.4000	469.0000
1.0	1	C$3	C$3	1.3700	520.0000
1.0	1	C$5	C$5	1.2040	590.0000
1.0	1	C$1	O$1	1.4250	386.0000
1.0	1	C$2	O$2	1.2500	280.0000
1.0	1	C$3	O$3	1.2300	300.0000
1.0	1	C$1	N$1	1.4490	337.0000
1.0	1	C$2	N$2	1.3810	427.0000
1.0	1	C$5	N$5	1.1580	649.0000
1.0	1	C$1	S$1	1.8100	222.0000
1.0	1	C$1	H$1	1.0900	331.0000

```
 1.0    1      O$1   O$1         1.4800     590.0000
 1.0    1      O$3   O$3         1.2080     590.0000
 1.0    1      O$1   N$1         1.2400     300.0000
 1.0    1      O$2   N$2         1.1900     450.0000
 1.0    1      O$3   N$3         1.1860     590.0000
 1.0    1      O$1   H$1         0.9600     553.0000
 1.0    1      N$1   N$1         1.1300     300.0000
 1.0    1      N$1   H$1         1.0100     434.0000
 1.0    1      S$1   S$1         2.0380     166.0000
 1.0    1      S$1   H$1         1.3360     274.0000
 1.0    1      O$1   P$1         1.6100     230.0000
 1.0    1      O$2   P$2         1.4800     525.0000
 1.0    1      P$1   H$1         1.5000     200.0000
#quadratic_angle    amber
> E = K2 * (Theta - Theta0)^2
!Ver  Ref     I     J     K        Theta0        K2
!---- ---    ----  ----  ----     --------     -------
 1.0   3     HW    OW    HW       104.5200     100.0000
 1.0   3     O     C     O        126.0000      80.0000
 1.0   3     C     CH    N3       109.7000      80.0000
 1.0   3     CH    CH    N3       109.7000      80.0000
 1.0   3     C     CT    N3       112.0000      80.0000
 1.0   3     CH    N3    H3       109.5000      35.0000
 1.0   3     CT    N3    CT       113.0000      50.0000
 1.0   3     P     OS    P        120.5000     100.0000
 1.0   1     C     C2    C2       112.4000      63.0000
 1.0   1     C     C2    CH       112.4000      63.0000
 1.0   1     C     C2    N        110.3000      80.0000
 1.0   1     C     C2    NT       111.2000      80.0000
 1.0   1     C     CA    CA       120.0000      85.0000
 1.0   1     C     CA    HC       120.0000      35.0000
 1.0   1     C     CB    CB       119.2000      85.0000
 1.0   1     C     CB    NB       130.0000      70.0000
 1.0   1     C     CD    CD       120.0000      85.0000
 1.0   1     C     CH    C2       111.1000      63.0000
 1.0   1     C     CH    C3       111.1000      63.0000
 1.0   1     C     CH    CH       111.1000      63.0000
 1.0   1     C     CH    N        110.1000      63.0000
 1.0   1     C     CH    NT       109.7000      80.0000
```

1.0	1	C	CJ	CJ	120.7000	85.0000
1.0	1	C	CM	C3	119.7000	85.0000
1.0	1	C	CM	CJ	120.7000	85.0000
1.0	1	C	CM	CM	120.7000	85.0000
1.0	1	C	CM	CT	119.7000	70.0000
1.0	1	C	CM	HC	119.7000	35.0000
1.0	1	C	CT	CT	111.1000	63.0000
1.0	1	C	CT	HC	109.5000	35.0000
1.0	1	C	CT	N	110.1000	63.0000
1.0	1	C	N	C2	121.9000	50.0000
1.0	1	C	N	C3	121.9000	50.0000
1.0	1	C	N	CH	121.9000	50.0000
1.0	1	C	N	CT	121.9000	50.0000
1.0	1	C	N	H	119.8000	35.0000
1.0	1	C	N	H2	120.0000	35.0000
1.0	1	C	N*	CH	117.6000	70.0000
1.0	1	C	N*	CJ	121.6000	70.0000
1.0	1	C	N*	CM	121.6000	70.0000
1.0	1	C	N*	CT	117.6000	70.0000
1.0	1	C	N*	H	119.2000	35.0000
1.0	1	C	NA	C	126.4000	70.0000
1.0	1	C	NA	CA	125.2000	70.0000
1.0	1	C	NA	H	116.8000	35.0000
1.0	1	C	NC	CA	120.5000	70.0000
1.0	1	C	OH	HO	113.0000	35.0000
1.0	1	C*	C2	CH	115.6000	63.0000
1.0	1	C*	CB	CA	134.9000	85.0000
1.0	1	C*	CB	CD	134.9000	85.0000
1.0	1	C*	CB	CN	108.8000	85.0000
1.0	1	C*	CG	NA	108.7000	70.0000
1.0	1	C*	CT	HC	109.5000	35.0000
1.0	1	C*	CW	HC	120.0000	35.0000
1.0	1	C*	CW	NA	108.7000	70.0000
1.0	1	C2	C	N	116.6000	70.0000
1.0	1	C2	C	O	120.4000	80.0000
1.0	1	C2	C	O2	117.0000	70.0000
1.0	1	C2	C*	CB	128.6000	70.0000
1.0	1	C2	C*	CG	125.0000	70.0000
1.0	1	C2	C*	CW	125.0000	70.0000

1.0	1	C2	C2	C2	112.4000	63.0000
1.0	1	C2	C2	CH	112.4000	63.0000
1.0	1	C2	C2	N	111.2000	80.0000
1.0	1	C2	C2	N2	111.2000	80.0000
1.0	1	C2	C2	N3	111.2000	80.0000
1.0	1	C2	C2	NT	111.2000	80.0000
1.0	1	C2	C2	OS	109.5000	80.0000
1.0	1	C2	C2	S	114.7000	50.0000
1.0	1	C2	CA	CA	120.0000	70.0000
1.0	1	C2	CA	CD	120.0000	70.0000
1.0	1	C2	CC	CF	131.9000	70.0000
1.0	1	C2	CC	CG	129.0000	70.0000
1.0	1	C2	CC	CV	131.9000	70.0000
1.0	1	C2	CC	CW	129.0000	70.0000
1.0	1	C2	CC	NA	122.2000	70.0000
1.0	1	C2	CC	NB	121.0000	70.0000
1.0	1	C2	CH	C3	111.5000	63.0000
1.0	1	C2	CH	CH	111.5000	63.0000
1.0	1	C2	CH	N	109.7000	80.0000
1.0	1	C2	CH	N*	109.5000	80.0000
1.0	1	C2	CH	NT	109.7000	80.0000
1.0	1	C2	CH	OH	109.5000	80.0000
1.0	1	C2	CH	OS	109.5000	80.0000
1.0	1	C2	N	CH	118.0000	50.0000
1.0	1	C2	N	H	118.4000	38.0000
1.0	1	C2	N2	CA	123.2000	50.0000
1.0	1	C2	N2	H2	118.4000	35.0000
1.0	1	C2	N2	H3	118.4000	35.0000
1.0	1	C2	N3	H3	109.5000	35.0000
1.0	1	C2	NT	H2	109.5000	35.0000
1.0	1	C2	OH	HO	108.5000	55.0000
1.0	1	C2	OS	C2	111.8000	100.0000
1.0	1	C2	OS	C3	111.8000	100.0000
1.0	1	C2	OS	HO	108.5000	55.0000
1.0	1	C2	OS	P	120.5000	100.0000
1.0	1	C2	S	C3	98.9000	62.0000
1.0	3	C2	S	LP	96.7000	150.0000
1.0	1	C2	S	S	103.7000	68.0000
1.0	1	C2	SH	HS	96.0000	44.0000

1.0	3	C2	SH	LP	96.7000	150.0000
1.0	1	C3	C	N	116.6000	70.0000
1.0	1	C3	C	O	120.4000	80.0000
1.0	1	C3	C	O2	117.0000	70.0000
1.0	1	C3	C2	CH	112.4000	63.0000
1.0	1	C3	C2	OS	109.5000	80.0000
1.0	1	C3	CH	C3	111.5000	63.0000
1.0	1	C3	CH	CH	111.5000	63.0000
1.0	1	C3	CH	N	109.5000	80.0000
1.0	1	C3	CH	NT	109.7000	80.0000
1.0	1	C3	CH	OH	109.5000	80.0000
1.0	1	C3	CM	CJ	119.7000	85.0000
1.0	1	C3	N	H	118.4000	38.0000
1.0	1	C3	N*	CB	125.8000	70.0000
1.0	1	C3	N*	CE	128.8000	70.0000
1.0	1	C3	N*	CK	128.8000	70.0000
1.0	1	C3	N2	CA	123.2000	50.0000
1.0	1	C3	N2	H2	118.4000	35.0000
1.0	1	C3	N3	H3	109.5000	35.0000
1.0	1	C3	OH	HO	108.5000	55.0000
1.0	1	C3	OS	P	120.5000	100.0000
1.0	3	C3	S	LP	96.7000	150.0000
1.0	1	C3	S	S	103.7000	68.0000
1.0	1	C3	SH	HS	96.0000	44.0000
1.0	3	C3	SH	LP	96.7000	150.0000
1.0	1	CA	C	CA	120.0000	85.0000
1.0	1	CA	C	OH	120.0000	70.0000
1.0	1	CT	C	OH	117.0000	70.0000
1.0	3	CT	C	O2	117.0000	70.0000
1.0	1	CA	C2	CH	114.0000	63.0000
1.0	1	CA	CA	CA	120.0000	85.0000
1.0	1	CA	CA	CB	120.0000	85.0000
1.0	1	CA	CA	CN	120.0000	85.0000
1.0	1	CA	CA	CT	120.0000	70.0000
1.0	1	CA	CA	HC	120.0000	35.0000
1.0	1	CA	CB	CB	117.3000	85.0000
1.0	1	CA	CB	CN	116.2000	85.0000
1.0	1	CA	CB	NB	132.4000	70.0000
1.0	1	CA	CD	CD	120.0000	85.0000

1.0	1	CA	CJ	CJ	117.0000	85.0000
1.0	1	CA	CM	CM	117.0000	85.0000
1.0	1	CA	CM	HC	123.3000	35.0000
1.0	1	CA	CN	CB	122.7000	85.0000
1.0	1	CA	CN	NA	132.8000	70.0000
1.0	1	CA	CT	CT	114.0000	63.0000
1.0	1	CA	CT	HC	109.5000	35.0000
1.0	1	CA	N2	CT	123.2000	50.0000
1.0	1	CA	N2	H	120.0000	35.0000
1.0	1	CA	N2	H2	120.0000	35.0000
1.0	1	CA	N2	H3	120.0000	35.0000
1.0	1	CA	NA	H	118.0000	35.0000
1.0	1	CA	NC	CB	112.2000	70.0000
1.0	1	CA	NC	CI	118.6000	70.0000
1.0	1	CA	NC	CQ	118.6000	70.0000
1.0	1	CB	C	NA	111.3000	70.0000
1.0	1	CB	C	O	128.8000	80.0000
1.0	1	CB	C*	CG	106.4000	85.0000
1.0	1	CB	C*	CT	128.6000	70.0000
1.0	1	CB	C*	CW	106.4000	85.0000
1.0	1	CB	C*	HC	126.8000	35.0000
1.0	1	CB	CA	HC	120.0000	35.0000
1.0	1	CB	CA	N2	123.5000	70.0000
1.0	1	CB	CA	NC	117.3000	70.0000
1.0	1	CB	CB	N*	106.2000	70.0000
1.0	1	CB	CB	NB	110.4000	70.0000
1.0	1	CB	CB	NC	127.7000	70.0000
1.0	1	CB	CD	CD	120.0000	85.0000
1.0	1	CB	CN	CD	122.7000	85.0000
1.0	1	CB	CN	NA	104.4000	70.0000
1.0	1	CB	N*	CE	105.4000	70.0000
1.0	1	CB	N*	CH	125.8000	70.0000
1.0	1	CB	N*	CK	105.4000	70.0000
1.0	1	CB	N*	CT	125.8000	70.0000
1.0	1	CB	N*	H	127.3000	35.0000
1.0	1	CB	NB	CE	103.8000	70.0000
1.0	1	CB	NB	CK	103.8000	70.0000
1.0	1	CB	NC	CI	111.0000	70.0000
1.0	1	CB	NC	CQ	111.0000	70.0000

1.0	1	CC	C2	CH	113.1000	63.0000
1.0	1	CC	CF	NB	109.9000	70.0000
1.0	1	CC	CG	NA	105.9000	70.0000
1.0	1	CC	CT	CT	113.1000	63.0000
1.0	1	CC	CT	HC	109.5000	35.0000
1.0	1	CC	CV	HC	120.0000	35.0000
1.0	1	CC	CV	NB	109.9000	70.0000
1.0	1	CC	CW	HC	120.0000	35.0000
1.0	1	CC	CW	NA	105.9000	70.0000
1.0	1	CC	NA	CP	107.3000	70.0000
1.0	1	CC	NA	CR	107.3000	70.0000
1.0	1	CC	NA	H	126.3000	35.0000
1.0	1	CC	NB	CP	105.3000	70.0000
1.0	1	CC	NB	CR	105.3000	70.0000
1.0	1	CD	C	CD	120.0000	85.0000
1.0	1	CD	C	OH	120.0000	70.0000
1.0	1	CD	CA	CD	120.0000	85.0000
1.0	1	CD	CB	CN	116.2000	85.0000
1.0	1	CD	CD	CD	120.0000	85.0000
1.0	1	CD	CD	CN	120.0000	85.0000
1.0	1	CD	CN	NA	132.8000	70.0000
1.0	1	CE	N*	CH	128.8000	70.0000
1.0	1	CE	N*	CT	128.8000	70.0000
1.0	1	CE	N*	H	127.3000	35.0000
1.0	1	CF	CC	NA	105.9000	70.0000
1.0	1	CF	NB	CP	105.3000	70.0000
1.0	1	CF	NB	CR	105.3000	70.0000
1.0	1	CG	CC	NA	108.7000	70.0000
1.0	1	CG	CC	NB	109.9000	70.0000
1.0	1	CG	NA	CN	111.6000	70.0000
1.0	1	CG	NA	CP	107.3000	70.0000
1.0	1	CG	NA	CR	107.3000	70.0000
1.0	1	CG	NA	H	126.3000	35.0000
1.0	1	CH	C	N	116.6000	70.0000
1.0	1	CH	C	O	120.4000	80.0000
1.0	1	CH	C	O2	117.0000	65.0000
1.0	1	CH	C	OH	115.0000	70.0000
1.0	1	CH	C2	CH	112.4000	63.0000
1.0	1	CH	C2	OH	109.5000	80.0000

1.0	1	CH	C2	OS	109.5000	80.0000
1.0	1	CH	C2	S	114.7000	50.0000
1.0	1	CH	C2	SH	108.6000	50.0000
1.0	1	CH	CH	CH	111.5000	63.0000
1.0	1	CH	CH	N	109.7000	80.0000
1.0	1	CH	CH	N*	109.5000	80.0000
1.0	1	CH	CH	NT	109.7000	80.0000
1.0	1	CH	CH	OH	109.5000	80.0000
1.0	1	CH	CH	OS	109.5000	80.0000
1.0	1	CH	N	H	118.4000	38.0000
1.0	1	CH	N*	CJ	121.2000	70.0000
1.0	1	CH	N*	CK	128.8000	70.0000
1.0	1	CH	NT	H2	109.5000	35.0000
1.0	1	CH	OH	HO	108.5000	55.0000
1.0	1	CH	OS	CH	111.8000	100.0000
1.0	1	CH	OS	HO	108.5000	55.0000
1.0	1	CH	OS	P	120.5000	100.0000
1.0	1	CJ	C	NA	114.1000	70.0000
1.0	1	CJ	C	O	125.3000	80.0000
1.0	1	CJ	CA	N2	120.1000	70.0000
1.0	1	CJ	CA	NC	121.5000	70.0000
1.0	1	CJ	CJ	N*	121.2000	70.0000
1.0	1	CJ	CM	CT	119.7000	85.0000
1.0	1	CJ	N*	CT	121.2000	70.0000
1.0	1	CJ	N*	H	119.2000	35.0000
1.0	1	CK	N*	CT	128.8000	70.0000
1.0	1	CM	C	NA	114.1000	70.0000
1.0	1	CM	C	O	125.3000	80.0000
1.0	1	CM	CA	N2	120.1000	70.0000
1.0	1	CM	CA	NC	121.5000	70.0000
1.0	1	CM	CJ	N*	121.2000	70.0000
1.0	1	CM	CM	CT	119.7000	70.0000
1.0	1	CM	CM	HC	119.7000	35.0000
1.0	1	CM	CM	N*	121.2000	70.0000
1.0	1	CM	CT	HC	109.5000	35.0000
1.0	1	CM	N*	CT	121.2000	70.0000
1.0	1	CM	N*	H	119.2000	35.0000
1.0	1	CN	CA	HC	120.0000	35.0000
1.0	1	CN	NA	CW	111.6000	70.0000

1.0	1	CN	NA	H	123.1000	35.0000
1.0	1	CP	NA	H	126.3000	35.0000
1.0	1	CR	NA	CW	107.3000	70.0000
1.0	1	CR	NA	H	126.3000	35.0000
1.0	1	CR	NB	CV	105.3000	70.0000
1.0	1	CT	C	N	116.6000	70.0000
1.0	1	CT	C	O	120.4000	80.0000
1.0	1	CT	C*	CW	125.0000	70.0000
1.0	1	CT	CC	CV	131.9000	70.0000
1.0	1	CT	CC	CW	129.0000	70.0000
1.0	1	CT	CC	NA	122.2000	70.0000
1.0	1	CT	CC	NB	121.0000	70.0000
1.0	1	CT	CT	CT	109.5000	40.0000
1.0	1	CT	CT	C*	115.6000	63.0000
1.0	1	CT	CT	HC	109.5000	35.0000
1.0	1	CT	CT	N	109.7000	80.0000
1.0	1	CT	CT	N*	109.5000	50.0000
1.0	1	CT	CT	N2	111.2000	80.0000
1.0	1	CT	CT	N3	111.2000	80.0000
1.0	1	CT	CT	OH	109.5000	50.0000
1.0	1	CT	CT	OS	109.5000	50.0000
1.0	1	CT	CT	S	114.7000	50.0000
1.0	1	CT	CT	SH	108.6000	50.0000
1.0	1	CT	N	CT	118.0000	50.0000
1.0	1	CT	N	H	118.4000	38.0000
1.0	1	CT	N2	H3	118.4000	35.0000
1.0	1	CT	N3	H3	109.5000	35.0000
1.0	1	CT	OH	HO	108.5000	55.0000
1.0	1	CT	OS	CT	109.5000	60.0000
1.0	1	CT	OS	P	120.5000	100.0000
1.0	1	CT	S	CT	98.9000	62.0000
1.0	3	CT	S	LP	96.7000	150.0000
1.0	1	CT	S	S	103.7000	68.0000
1.0	1	CT	SH	HS	96.0000	44.0000
1.0	3	CT	SH	LP	96.7000	150.0000
1.0	1	CV	CC	NA	105.9000	70.0000
1.0	1	CW	C*	HC	126.8000	35.0000
1.0	1	CW	CC	NA	108.7000	70.0000
1.0	1	CW	CC	NB	109.9000	70.0000

1.0	1	CW	NA	H	125.3000	35.0000
1.0	1	H	N	H	120.0000	35.0000
1.0	1	H2	N2	H2	120.0000	35.0000
1.0	1	H2	NT	H2	109.5000	35.0000
1.0	1	H3	N	H3	120.0000	35.0000
1.0	1	H3	N2	H3	120.0000	35.0000
1.0	1	H3	N3	H3	109.5000	35.0000
1.0	1	HC	CK	N*	123.0000	35.0000
1.0	1	HC	CK	NB	123.0000	35.0000
1.0	1	HC	CM	N*	119.1000	35.0000
1.0	1	HC	CQ	NC	115.4000	35.0000
1.0	1	HC	CR	NA	120.0000	35.0000
1.0	1	HC	CR	NB	120.0000	35.0000
1.0	1	HC	CT	HC	109.5000	35.5000
1.0	1	HC	CT	N	109.5000	38.0000
1.0	1	HC	CT	N*	109.5000	35.0000
1.0	1	HC	CT	N2	109.5000	35.0000
1.0	1	HC	CT	N3	109.5000	35.0000
1.0	1	HC	CT	OH	109.5000	35.0000
1.0	1	HC	CT	OS	109.5000	35.0000
1.0	1	HC	CT	S	109.5000	35.0000
1.0	1	HC	CT	SH	109.5000	35.0000
1.0	1	HC	CV	NB	120.0000	35.0000
1.0	1	HC	CW	NA	120.0000	35.0000
1.0	1	HO	OH	HO	104.5000	47.0000
1.0	1	HO	OH	P	108.5000	45.0000
1.0	1	HS	SH	HS	92.1000	35.0000
1.0	3	HS	SH	LP	96.7000	150.0000
1.0	3	LP	S	LP	160.0000	150.0000
1.0	3	LP	S	S	96.7000	150.0000
1.0	3	LP	SH	LP	160.0000	150.0000
1.0	1	N	C	O	122.9000	80.0000
1.0	1	N*	C	NA	115.4000	70.0000
1.0	1	N*	C	NC	118.6000	70.0000
1.0	1	N*	C	O	120.9000	80.0000
1.0	1	N*	CB	NC	126.2000	70.0000
1.0	1	N*	CE	NB	113.9000	70.0000
1.0	1	N*	CH	OS	109.5000	80.0000
1.0	1	N*	CK	NB	113.9000	70.0000

1.0	1	N*	CT	OS	109.5000	50.0000
1.0	1	N2	CA	N2	120.0000	70.0000
1.0	1	N2	CA	NA	116.0000	70.0000
1.0	1	N2	CA	NC	119.3000	70.0000
1.0	1	NA	C	O	120.6000	80.0000
1.0	1	NA	CA	NC	123.3000	70.0000
1.0	1	NA	CP	NA	110.7000	70.0000
1.0	1	NA	CP	NB	111.6000	70.0000
1.0	1	NA	CR	NA	110.7000	70.0000
1.0	1	NA	CR	NB	111.6000	70.0000
1.0	1	NC	C	O	122.5000	80.0000
1.0	1	NC	CI	NC	129.1000	70.0000
1.0	1	NC	CQ	NC	129.1000	70.0000
1.0	1	O	C	O2	126.0000	80.0000
1.0	1	O	C	OH	126.0000	80.0000
1.0	1	O2	C	O2	126.0000	80.0000
1.0	1	O2	P	O2	119.9000	140.0000
1.0	1	O2	P	OH	108.2000	45.0000
1.0	1	O2	P	OS	108.2000	100.0000
1.0	1	OH	P	OS	102.6000	45.0000
1.0	1	OS	P	OS	102.6000	45.0000
1.1	4	HO	OH	HO	104.5000	47.0000
1.1	4	CS	OT	HY	109.3500	53.6000
1.1	4	AC	OA	HY	109.3500	53.6000
1.1	4	BC	OB	HY	109.3500	53.6000
1.1	4	CS	OT	CS	117.0000	60.0000
1.1	4	AC	OA	CS	115.0000	62.0000
1.1	4	BC	OB	CS	116.4000	62.0000
1.1	4	CS	OE	AC	113.8000	90.7000
1.1	4	CS	OE	BC	111.9000	90.7000
1.1	4	HT	CS	HT	107.8500	33.6000
1.1	4	AH	AC	HT	107.8500	33.6000
1.1	4	BH	BC	HT	107.8500	33.6000
1.1	4	HT	CS	CS	108.7200	43.0000
1.1	4	HC	CT	CS	108.7200	43.0000
1.1	4	HT	CS	CT	108.7200	43.0000
1.1	4	AH	AC	CS	108.7200	43.0000
1.1	4	BH	BC	CS	108.7200	43.0000
1.1	4	HT	CS	AC	108.7200	43.0000

1.1	4	HT	CS	BC	108.7200	43.0000
1.1	4	HT	CS	OT	109.8900	45.9000
1.1	4	AH	AC	OA	109.8900	45.9000
1.1	4	BH	BC	OB	109.8900	45.9000
1.1	4	HT	AC	OA	109.8900	45.9000
1.1	4	HT	BC	OB	109.8900	45.9000
1.1	4	HT	CS	OA	109.8900	45.9000
1.1	4	HT	CS	OB	109.8900	45.9000
1.1	4	HT	CS	OE	107.2400	45.2000
1.1	4	HT	CS	C	109.5000	35.0000
1.1	4	AH	AC	OE	107.2400	45.2000
1.1	4	BH	BC	OE	107.2400	45.2000
1.1	4	HT	AC	OE	107.2400	45.2000
1.1	4	HT	BC	OE	107.2400	45.2000
1.1	4	CS	CS	CS	110.7000	38.0000
1.1	4	CS	CS	CT	110.7000	38.0000
1.1	4	CS	CS	AC	110.7000	38.0000
1.1	4	CS	CS	BC	110.7000	38.0000
1.1	4	CS	CS	OT	110.1000	75.7000
1.1	4	CS	CT	OH	110.1000	75.7000
1.1	4	CS	CS	OA	110.1000	75.7000
1.1	4	CS	CS	OB	110.1000	75.7000
1.1	4	CS	C	O	120.4000	80.0000
1.1	4	AC	CS	OT	110.1000	75.7000
1.1	4	BC	CS	OT	110.1000	75.7000
1.1	4	BC	CS	OB	110.1000	75.7000
1.1	4	BC	CS	OA	110.1000	75.7000
1.1	4	AC	CS	OB	110.1000	75.7000
1.1	4	AC	CS	OA	110.1000	75.7000
1.1	4	CS	AC	OA	110.1000	75.7000
1.1	4	CS	BC	OB	110.1000	75.7000
1.1	4	CS	CS	OE	109.4000	81.0000
1.1	4	CT	CS	OE	109.4000	81.0000
1.1	4	CS	AC	OE	109.4000	81.0000
1.1	4	CS	BC	OE	109.4000	81.0000
1.1	4	CS	OE	CS	113.8000	90.7000
1.1	4	OE	CS	OT	111.5500	92.6000
1.1	4	OE	AC	OA	111.5500	92.6000
1.1	4	OE	BC	OB	107.4000	92.6000

```
 1.1    4      BC     CS     N         109.7000      80.0000
 1.1    4      CS     CS     N         109.7000      80.0000
 1.1    4      HT     CS     N         109.5000      38.0000
 1.1    4      CS     N      H         118.4000      38.0000
 1.1    4      CS     N      C         121.9000      50.0000
 1.1    4      C      N      H         119.8000      35.0000
 1.1    4      N      C      O         122.9000      80.0000
 1.1    4      N      C      CS        116.6000      70.0000
 1.0    1      $$     C$4    $$        109.5000      63.0000
 1.0    1      $$     C$3    $$        120.0000      85.0000
 1.0    1      $$     C$2    $$        180.0000     200.0000
 1.0    1      $$     O$2    $$        109.5000     100.0000
 1.0    1      $$     N$4    $$        109.5000      60.0000
 1.0    1      $$     N$3    $$        114.0000      60.0000
 1.0    1      $$     N$2    $$        120.0000      60.0000
 1.0    1      $$     S$2    $$        109.5000      60.0000
 1.0    1      $$     P$4    $$        109.5000     110.0000
 1.0    1      C$$    S$2    H$$        96.0000      44.0000
 1.0    1      C$$    S$2    C$$        99.0000      62.0000
 1.0    1      C$$    S$2    S$$        96.0000      44.0000
#torsion_3      amber
> E = SUM(n=1,3) { V(n) * [ 1 + cos(n*Phi - Phi0(n)) ] }

!Ver   Ref      I      J      K      L            V1       Phi0
V2       Phi0          V3      Phi0
!---- ---      ----   ----   ----   ----         -------   ------
-------  ------        -------  ------
 1.0    3       *      CB     CD     *            0.0000      0.0
5.3000  180.0         0.0000     0.0
 1.0    1       *      C      C2     *            0.0000      0.0
0.0000     0.0        0.0000  180.0
 1.0    1       *      C      CA     *            0.0000      0.0
5.3000  180.0         0.0000     0.0
 1.0    1       *      C      CB     *            0.0000      0.0
4.4000  180.0         0.0000     0.0
 1.0    1       *      C      CD     *            0.0000      0.0
5.3000  180.0         0.0000     0.0
 1.0    1       *      C      CH     *            0.0000      0.0
0.0000     0.0        0.0000     0.0
```

```
 1.0    1      *      C     CJ     *           0.0000    0.0
3.1000  180.0       0.0000    0.0
 1.0    1      *      C     CM     *           0.0000    0.0
3.1000  180.0       0.0000    0.0
 1.0    1      *      C     CT     *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C     N      *           0.0000    0.0
10.0000  180.0      0.0000    0.0
 1.0    1      *      C     N*     *           0.0000    0.0
5.8000  180.0       0.0000    0.0
 1.0    1      *      C     NA     *           0.0000    0.0
5.4000  180.0       0.0000    0.0
 1.0    1      *      C     NC     *           0.0000    0.0
8.0000  180.0       0.0000    0.0
 1.0    1      *      C     OH     *           0.0000    0.0
1.8000  180.0       0.0000    0.0
 1.0    1      *      C*    C2     *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C*    CB     *           0.0000    0.0
4.8000  180.0       0.0000    0.0
 1.0    1      *      C*    CG     *           0.0000    0.0
23.6000  180.0      0.0000    0.0
 1.0    1      *      C*    CT     *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C*    CW     *           0.0000    0.0
23.6000  180.0      0.0000    0.0
 1.0    1      *      C2    C2     *           0.0000    0.0
0.0000    0.0       2.0000    0.0
 1.0    1      *      C2    CA     *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2    CC     *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2    CH     *           0.0000    0.0
0.0000    0.0       2.0000    0.0
 1.0    1      *      C2    N      *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2    N2     *           0.0000    0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2    N3     *           0.0000    0.0
```

```
0.0000     0.0       1.4000    0.0
 1.0    1       *     C2     NT      *                0.0000      0.0
0.0000     0.0       1.0000    0.0
 1.0    1       *     C2     OH      *                0.0000      0.0
0.0000     0.0       0.5000    0.0
 1.0    1       *     C2     OS      *                0.0000      0.0
0.0000     0.0       1.4500    0.0
 1.0    1       *     C2     S       *                0.0000      0.0
0.0000     0.0       1.0000    0.0
 1.0    1       *     C2     SH      *                0.0000      0.0
0.0000     0.0       0.7500    0.0
 1.0    1       *     CA     CA      *                0.0000      0.0
5.3000   180.0       0.0000    0.0
 1.0    1       *      CA     CB      *                0.0000      0.0
10.2000  180.0       0.0000    0.0
 1.0    1       *     CA     CD      *                0.0000      0.0
5.3000   180.0       0.0000    0.0
 1.0    1       *     CA     CJ      *                0.0000      0.0
3.7000   180.0       0.0000    0.0
 1.0    1       *     CA     CM      *                0.0000      0.0
3.7000   180.0       0.0000    0.0
 1.0    1       *      CA     CN      *                0.0000      0.0
10.6000  180.0       0.0000    0.0
 1.0    1       *     CA     CT      *                0.0000      0.0
0.0000     0.0       0.0000    0.0
 1.0    1       *     CA     N2      *                0.0000      0.0
6.8000   180.0       0.0000    0.0
 1.0    1       *     CA     NA      *                0.0000      0.0
6.0000   180.0       0.0000    0.0
 1.0    1       *     CA     NC      *                0.0000      0.0
9.6000   180.0       0.0000    0.0
 1.0    1       *      CB     CB      *                0.0000      0.0
16.3000  180.0       0.0000    0.0
 1.0    1       *      CB     CN      *                0.0000      0.0
20.0000  180.0       0.0000    0.0
 1.0    1       *     CB     N*      *                0.0000      0.0
6.6000   180.0       0.0000    0.0
 1.0    1       *     CB     NB      *                0.0000      0.0
5.1000   180.0       0.0000    0.0
```

```
 1.0   3      *      CB    NC     *            0.0000     0.0
8.3000  180.0       0.0000    0.0
 1.0   1      *      CC    CF     *            0.0000     0.0
14.3000  180.0      0.0000    0.0
 1.0   1      *      CC    CG     *            0.0000     0.0
15.9000  180.0      0.0000    0.0
 1.0   1      *      CC    CT     *            0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      CC    CV     *            0.0000     0.0
14.3000  180.0      0.0000    0.0
 1.0   1      *      CC    CW     *            0.0000     0.0
15.9000  180.0      0.0000    0.0
 1.0   1      *      CC    NA     *            0.0000     0.0
5.6000  180.0       0.0000    0.0
 1.0   1      *      CC    NB     *            0.0000     0.0
4.8000  180.0       0.0000    0.0
 1.0   1      *      CD    CD     *            0.0000     0.0
5.3000  180.0       0.0000    0.0
 1.0   1      *      CD    CN     *            0.0000     0.0
5.3000  180.0       0.0000    0.0
 1.0   1      *      CE    N*     *            0.0000     0.0
6.7000  180.0       0.0000    0.0
 1.0   1      *      CE    NB     *            0.0000     0.0
20.0000  180.0      0.0000    0.0
 1.0   1      *      CF    NB     *            0.0000     0.0
4.8000  180.0       0.0000    0.0
 1.0   1      *      CG    NA     *            0.0000     0.0
6.0000  180.0       0.0000    0.0
 1.0   1      *      CH    CH     *            0.0000     0.0
0.0000    0.0       2.0000    0.0
 1.0   1      *      CH    N      *            0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      CH    N*     *            0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0   1      *      CH    NT     *            0.0000     0.0
0.0000    0.0       1.0000    0.0
 1.0   1      *      CH    OH     *            0.0000     0.0
0.0000    0.0       0.5000    0.0
 1.0   1      *      CH    OS     *            0.0000     0.0
```

```
0.0000    0.0       1.4500    0.0
 1.0    1       *      CI     NC      *            0.0000     0.0
13.5000  180.0      0.0000    0.0
 1.0    1       *      CJ     CJ      *            0.0000     0.0
24.4000  180.0      0.0000    0.0
 1.0    1       *      CJ     CM      *            0.0000     0.0
24.4000  180.0      0.0000    0.0
 1.0    1       *      CJ     N*      *            0.0000     0.0
7.4000  180.0       0.0000    0.0
 1.0    1       *      CK     N*      *            0.0000     0.0
6.7000  180.0       0.0000    0.0
 1.0    1       *      CK     NB      *            0.0000     0.0
20.0000  180.0      0.0000    0.0
 1.0    1       *      CM     CM      *            0.0000     0.0
24.4000  180.0      0.0000    0.0
 1.0    1       *      CM     CT      *            0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1       *      CM     N*      *            0.0000     0.0
7.4000  180.0       0.0000    0.0
 1.0    1       *      CN     NA      *            0.0000     0.0
12.2000  180.0      0.0000    0.0
 1.0    1       *      CP     NA      *            0.0000     0.0
9.3000  180.0       0.0000    0.0
 1.0    1       *      CP     NB      *            0.0000     0.0
10.0000  180.0      0.0000    0.0
 1.0    1       *      CQ     NC      *            0.0000     0.0
13.5000  180.0      0.0000    0.0
 1.0    1       *      CR     NA      *            0.0000     0.0
9.3000  180.0       0.0000    0.0
 1.0    1       *      CR     NB      *            0.0000     0.0
10.0000  180.0      0.0000    0.0
 1.0    1       *      CT     CT      *            0.0000     0.0
0.0000    0.0       1.3000    0.0
 1.0    1       *      CT     N       *            0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1       *      CT     N*      *            0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1       *      CT     N2      *            0.0000     0.0
0.0000    0.0       0.0000    0.0
```

```
 1.0    1      *      CT     N3     *                0.0000     0.0
0.0000     0.0        1.4000     0.0
 1.0    1      *      CT     OH     *                0.0000     0.0
0.0000     0.0        0.5000     0.0
 1.0    1      *      CT     OS     *                0.0000     0.0
0.0000     0.0        1.1500     0.0
 1.0    1      *      CT     S      *                0.0000     0.0
0.0000     0.0        1.0000     0.0
 1.0    1      *      CT     SH     *                0.0000     0.0
0.0000     0.0        0.7500     0.0
 1.0    1      *      CV     NB     *                0.0000     0.0
4.8000   180.0        0.0000     0.0
 1.0    1      *      CW     NA     *                0.0000     0.0
6.0000   180.0        0.0000     0.0
 1.0    1      *      OH     P      *                0.0000     0.0
0.0000     0.0        0.7500     0.0
 1.0    1      *      OS     P      *                0.0000     0.0
0.0000     0.0        0.7500     0.0
 1.0    1      O      C      C2     N                0.0000     0.0
0.0000     0.0        0.2000   180.0
 1.0    1      O      C      CH     C2               0.0000     0.0
0.0000     0.0        0.1000   180.0
 1.0    1      O      C      CH     N                0.0000     0.0
0.0000     0.0        0.1000   180.0
 1.0    1      O      C      CH     CH               0.0000     0.0
0.0000     0.0        0.1000   180.0
 1.0    1      OS     C2     C2     OH               0.0000     0.0
0.5000     0.0        2.0000     0.0
 1.0    2      OH     C2     C2     OH               0.0000     0.0
0.5000     0.0        2.0000     0.0
 1.0    1      OS     C2     C2     OS               0.0000     0.0
0.5000     0.0        2.0000     0.0
 1.0    1      OS     C2     CH     OS               0.0000     0.0
0.5000     0.0        1.0000     0.0
 1.0    1      OS     C2     CH     OH               0.0000     0.0
0.5000     0.0        1.0000     0.0
 1.0    1      OH     C2     CH     OH               0.0000     0.0
0.5000     0.0        1.0000     0.0
 1.0    1      C2     C2     S      LP               0.0000     0.0
```

```
0.0000    0.0       0.0000    0.0
 1.0    1      CH     C2     SH     LP           0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      OS     CH     C2     OH           0.0000     0.0
0.5000    0.0       1.0000    0.0
 1.0    1      OH     CH     CH     OH           0.0000     0.0
0.5000    0.0       0.5000    0.0
 1.0    1      OS     CH     CH     OH           0.0000     0.0
0.5000    0.0       0.5000    0.0
 1.0    1      OS     CH     CH     OS           0.0000     0.0
0.5000    0.0       0.5000    0.0
 1.0    1      HC     CM     CM     CT           0.0000     0.0
1.7100  180.0       0.0000    0.0
 1.0    1      C      CM     CM     HC           0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0    1      N*     CM     CM     CT           0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0    1      CA     CM     CM     HC           0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0    1      N*     CM     CM     CA           0.0000     0.0
9.5100  180.0       0.0000    0.0
 1.0    1      HC     CM     CM     HC           0.0000     0.0
1.7100  180.0       0.0000    0.0
 1.0    1      N*     CM     CM     C            0.0000     0.0
9.5100  180.0       0.0000    0.0
 1.0    1      N*     CM     CM     HC           0.0000     0.0
6.5900  180.0       0.0000    0.0
 1.0    1      N      CT     C      O            0.0000     0.0
0.0000    0.0       0.0670  180.0
 1.0    1      HC     CT     C      O            0.0000     0.0
0.0000    0.0       0.0670  180.0
 1.0    1      CT     CT     C      O            0.0000     0.0
0.0000    0.0       0.0670  180.0
 1.0    1      CT     OS     CT     CT           0.0000     0.0
0.2000  180.0       0.3830    0.0
 1.0    1      OS     CT     CT     OS           0.0000     0.0
0.5000    0.0       0.1440    0.0
 1.0    1      OS     CT     CT     OH           0.0000     0.0
0.5000    0.0       0.1440    0.0
```

```
 1.0    1      OH     CT     CT     OH             0.0000     0.0
0.5000    0.0        0.1440     0.0
 1.0    1      H      N      C      O              0.6500     0.0
2.5000  180.0        0.0000     0.0
 1.0    1      C2     OS     C2     C3             0.0000     0.0
0.1000    0.0        0.7250     0.0
 1.0    1      C2     OS     C2     C2             0.0000     0.0
0.1000    0.0        1.4500     0.0
 1.0    1      C3     OS     C2     C3             0.0000     0.0
0.1000    0.0        1.4500     0.0
 1.0    1      CH     OS     CH     C2             0.0000     0.0
0.1000    0.0        0.7250     0.0
 1.0    1      CH     OS     CH     CH             0.0000     0.0
0.1000    0.0        0.7250     0.0
 1.0    1      C2     OS     CH     C2             0.0000     0.0
0.1000    0.0        0.7250     0.0
 1.0    1      C3     OS     CH     C3             0.0000     0.0
0.1000    0.0        0.7250     0.0
 1.0    1      CH     OS     CH     N*             0.0000     0.0
0.0000    0.0        0.7250     0.0
 1.0    1      C2     OS     CH     C3             0.0000     0.0
0.1000    0.0        0.7250     0.0
 1.0    1      OH     P      OS     C3             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OS     P      OS     C2             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OH     P      OS     C2             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OS     P      OS     CT             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OS     P      OS     CH             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OS     P      OS     C3             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OH     P      OS     CH             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      OH     P      OS     CT             0.0000     0.0
0.7500    0.0        0.2500     0.0
 1.0    1      LP     S      S      LP             0.0000     0.0
```

```
0.0000     0.0       0.0000     0.0
  1.0    1      LP     S      S      C2             0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.0    1      C2     S      S      C2             0.0000     0.0
3.5000     0.0       0.6000     0.0
  1.0    1      CT     S      S      CT             0.0000     0.0
3.5000     0.0       0.6000     0.0
  1.0    1      LP     S      S      CT             0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.1    4      *      CS     CS     *              0.0000     0.0
0.0000     0.0       1.0210     0.0
  1.1    4      *      CS     CT     *              0.0000     0.0
0.0000     0.0       1.0210     0.0
  1.1    4      *      AC     CS     *              0.0000     0.0
0.0000     0.0       1.0210     0.0
  1.1    4      *      BC     CS     *              0.0000     0.0
0.0000     0.0       1.0210     0.0
  1.1    4      *      CS     OT     *              0.0000     0.0
0.0000     0.0       0.4430     0.0
  1.1    4      *      CS     OE     *              0.0000     0.0
0.0000     0.0       0.9280     0.0
  1.1    4      *      AC     OE     *              0.0000     0.0
0.0000     0.0       0.9280     0.0
  1.1    4      *      BC     OE     *              0.0000     0.0
0.0000     0.0       0.9280     0.0
  1.1    4      *      AC     OA     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.1    4      *      BC     OB     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.1    4      *      CS     OA     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.1    4      *      CS     OB     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.1    4      *      CS     N      *              0.0000     0.0
0.0000     0.0       0.0000     0.0
  1.1    4      *      C      N      *              0.0000     0.0
10.0000  180.0       0.0000     0.0
  1.1    4      *      C      CS     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
```

```
 1.1    4      OE     AC     OA     CS             2.1500  300.0
0.0000    0.0       0.0000    0.0
 1.1    4      AH     AC     OA     CS             0.0000    0.0
1.7500   60.0       0.0000    0.0
 1.1    4      CS     AC     OA     CS             0.0000    0.0
0.0000    0.0       0.8500    0.0
 1.1    4      OE     AC     OA     HY             2.1500  300.0
0.0000    0.0       0.0000    0.0
 1.1    4      AH     AC     OA     HY             0.0000    0.0
1.7500   60.0       0.0000    0.0
 1.1    4      CS     AC     OA     HY             0.0000    0.0
0.0000    0.0       0.8500    0.0
 1.1    4      OE     BC     OB     CS            -1.0500    0.0
0.0000    0.0       0.0000    0.0
 1.1    4      BH     BC     OB     CS             0.0000    0.0
1.2500  240.0       0.0000    0.0
 1.1    4      CS     BC     OB     CS             0.0000    0.0
0.0000    0.0       1.4000    0.0
 1.1    4      OE     BC     OB     HY            -1.0500    0.0
0.0000    0.0       0.0000    0.0
 1.1    4      BH     BC     OB     HY             0.0000    0.0
1.2500  240.0       0.0000    0.0
 1.1    4      CS     BC     OB     HY             0.0000    0.0
0.0000    0.0       1.4000    0.0
 1.1    4      HT     AC     OA     CS             0.0000    0.0
0.0000    0.0       0.8500    0.0
 1.1    4      HT     BC     OB     CS             0.0000    0.0
0.0000    0.0       1.4000    0.0
 1.1    4      H      N      C      O              0.6500    0.0
2.5000  180.0       0.0000    0.0
 1.1    4      HT     CS     C      O              0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0    1      $$     C$1    C$1    $$             0.0000    0.0
0.0000    0.0       1.3000    0.0
 1.0    1      $$     C$2    C$2    $$             0.0000    0.0
5.3000  180.0       0.0000    0.0
 1.0    1      $$     C$3    C$3    $$             0.0000    0.0
16.3000  180.0       0.0000    0.0
 1.0    1      $$     C$5    C$5    $$             0.0000    0.0
```

```
0.0000  180.0      0.0000     0.0
 1.0    1       $$     C$1    O$1    $$           0.0000      0.0
0.0000     0.0      1.1000     0.0
 1.0    1       $$     C$1    N$1    $$           0.0000      0.0
0.0000     0.0      0.3000     0.0
 1.0    1       $$     C$2    N$2    $$           0.0000      0.0
5.8000  180.0      0.0000     0.0
 1.0    1       $$     C$3    N$3    $$            0.0000      0.0
10.0000  180.0      0.0000     0.0
 1.0    1       $$     C$1    S$1    $$           0.0000      0.0
0.0000     0.0      0.7500     0.0
 1.0    1       $$     S$1    S$1    $$           0.0000      0.0
3.5000     0.0      0.6000     0.0
 1.0    1       $$     O$1    O$1    $$           0.0000      0.0
0.0000     0.0      1.1000     0.0
 1.0    1       $$     O$1    N$1    $$           0.0000      0.0
0.0000     0.0      1.1000     0.0
 1.0    1       $$     O$1    P$1    $$           0.0000      0.0
0.0000     0.0      0.7500     0.0
 1.0    1       $$     N$1    N$1    $$           0.0000      0.0
0.0000     0.0      0.3000     0.0
#out_of_plane  amber
> E = Kchi * [ 1 + cos(n*Chi - Chi0) ]
!Ver  Ref      I     J     K     L          Kchi         n
    Chi0
!---- ---     ----  ----  ----  ----       -------      ------
 -------
 1.0   3      C*    NA    CA    CA          0.0000        2
180.0000
 1.0   1      N3    C     CH    C2          7.0000        3
180.0000
 1.0   1      C3    CA    CH    C3          7.0000        3
180.0000
 1.0   1      C     NT    CH    C3         14.0000        3
180.0000
 1.0   1      N3    C     CH    CH          7.0000        3
180.0000
 1.0   1      H2    N2    CH    H2          0.0000        3
180.0000
```

```
  1.0    1    *    CH   C2   *      14.0000    3
180.0000
  1.0    1    *    CH   CH   *      14.0000    3
180.0000
  1.0    1    *    CC   CC   *       0.0000    2
180.0000
  1.0    1    *    CC   CB   *       0.0000    2
180.0000
  1.0    1    C    N    CH   *      14.0000    3
180.0000
  1.0    1    C2   N    CH   *       1.0000    2
180.0000
  1.0    1    CT   N    CT   *       1.0000    2
180.0000
  1.0    1    H2   N    H2   *       1.0000    2
180.0000
  1.0    1    N2   CA   N2   *      10.5000    2
180.0000
  1.0    1    O2   C    O2   *      10.5000    2
180.0000
  1.0    1    C    NT   CH   *      14.0000    3
180.0000
  1.0    1    C    N3   CH   *      14.0000    3
180.0000
  1.0    1    O    C    *    *      10.5000    2
180.0000
  1.0    1    HC   C*   *    *       0.0000    2
180.0000
  1.0    1    HC   CW   *    *       0.0000    2
180.0000
  1.0    1    CB   CN   *    *       0.0000    2
180.0000
  1.0    1    CN   CB   *    *       0.0000    2
180.0000
  1.0    1    C*   CB   *    *       0.0000    2
180.0000
  1.0    1    CA   CB   *    *       0.0000    2
180.0000
  1.0    1    CA   CN   *    *       0.0000    2
```

```
180.0000
 1.0    1      NA     CN      *       *            0.0000          2
180.0000
 1.0    1      HC     CA      *       *            2.0000          2
180.0000
 1.0    1      H      N       *       *            1.0000          2
180.0000
 1.0    1      H2     N2      *       *            1.0000          2
180.0000
 1.0    1      H3     N2      *       *            1.0000          2
180.0000
 1.0    1      H2     NT      *       *            1.0000          2
180.0000
 1.0    1      H      NA      *       *            1.0000          2
180.0000
 1.0    1      $$     $$      $$      $$          10.0000          2
180.0000
#nonbond(12-6) amber
@type r-eps
@combination arithmetic
> E = EPSij * { (Rij*/Rij)^12 - 2(Rij*/Rij)^6 }
> where  EPSij = sqrt( EPSi * EPSj)
>           Rij* = (Ri* + Rj*)/2
!Ver  Ref      I            Ri*            EPSi
!---- ---     ----      -----------    -----------
 1.0    3      IM           5.0000         0.10000
 1.0    3      CU           2.4000         0.05000
 1.0    3      I            4.8000         0.40000
 1.0    3      OW           3.5360         0.15200
 1.0    3      MG           2.3400         0.10000
 1.0    3      C0           3.2000         0.10000
 1.0    3      QC           6.8000         0.00008
 1.0    3      QK           5.3200         0.00033
 1.0    3      QL           2.2800         0.01800
 1.0    3      QN           3.7400         0.00280
 1.0    3      QR           5.9200         0.00017
 1.0    1      C            3.7000         0.12000
 1.0    1      C*           3.7000         0.12000
 1.0    1      C2           3.8400         0.12000
```

1.0	1	C3	4.0000	0.15000
1.0	1	CA	3.7000	0.12000
1.0	1	CB	3.7000	0.12000
1.0	1	CC	3.7000	0.12000
1.0	1	CD	3.7000	0.12000
1.0	1	CE	3.7000	0.12000
1.0	1	CF	3.7000	0.12000
1.0	1	CG	3.7000	0.12000
1.0	1	CH	3.7000	0.09000
1.0	1	CI	3.7000	0.12000
1.0	1	CJ	3.7000	0.12000
1.0	1	CK	3.7000	0.12000
1.0	1	CM	3.7000	0.12000
1.0	1	CN	3.7000	0.12000
1.0	1	CP	3.7000	0.12000
1.0	1	CQ	3.7000	0.12000
1.0	1	CR	3.7000	0.12000
1.0	1	CT	3.6000	0.06000
1.0	1	CV	3.7000	0.12000
1.0	1	CW	3.7000	0.12000
1.0	1	H	2.0000	0.02000
1.0	1	H2	2.0000	0.02000
1.0	1	H3	2.0000	0.02000
1.0	1	HC	3.0800	0.01000
1.0	1	HO	2.0000	0.02000
1.0	1	HS	2.0000	0.02000
1.0	1	LP	2.4000	0.01600
1.0	1	N	3.5000	0.16000
1.0	1	N*	3.5000	0.16000
1.0	1	N2	3.5000	0.16000
1.0	1	N3	3.7000	0.08000
1.0	1	NA	3.5000	0.16000
1.0	1	NB	3.5000	0.16000
1.0	1	NC	3.5000	0.16000
1.0	1	NP	3.5000	0.16000
1.0	1	NT	3.7000	0.12000
1.0	1	O	3.2000	0.20000
1.0	1	O2	3.2000	0.20000
1.0	1	OH	3.3000	0.15000

```
 1.0    1     OS            3.3000        0.15000
 1.0    1     P             4.2000        0.20000
 1.0    1     S             4.0000        0.20000
 1.0    1     SH            4.0000        0.20000
 1.1    4     CS            3.6000        0.09030
 1.1    4     AC            3.6000        0.09030
 1.1    4     BC            3.6000        0.09030
 1.1    4     C             3.7000        0.12000
 1.1    4     H             2.0000        0.02000
 1.1    4     HY            1.6000        0.04980
 1.1    4     HT            2.9360        0.00450
 1.1    4     HO            2.0000        0.02000
 1.1    4     AH            2.9360        0.00450
 1.1    4     BH            2.9360        0.00450
 1.1    4     OT            3.2000        0.15910
 1.1    4     OA            3.2000        0.15910
 1.1    4     OB            3.2000        0.15910
 1.1    4     OE            3.2000        0.15910
 1.1    4     OH            3.3000        0.15000
 1.1    4     O             3.2000        0.20000
 1.1    4     N             3.5000        0.16000
#hydrogen_bond(10-12)    amber
> E = Aij/r^12 - Bij/r^10
!Ver  Ref     I    J          A              B
!---- ---    ---- ----   -----------    -----------
 1.0   3      H    OS       7557.0000      2385.0000
 1.0   3      H    OW       7557.0000      2385.0000
 1.0   3      H2   OS       7557.0000      2385.0000
 1.0   3      H2   OW       7557.0000      2385.0000
 1.0   3      HW   NB       7557.0000      2385.0000
 1.0   3      HW   NC      10238.0000      3071.0000
 1.0   3      HW   O        7557.0000      2385.0000
 1.0   3      HW   O2       4019.0000      1409.0000
 1.0   3      HW   OH       7557.0000      2385.0000
 1.0   3      HW   OS       7557.0000      2385.0000
 1.0   3      HW   S      265720.0000     35429.0000
 1.0   3      HW   SH     265720.0000     35429.0000
 1.0   1      H    NB       7557.0000      2385.0000
 1.0   1      H    NC      10238.0000      3071.0000
 1.0    1      H     O2          4019.0000      1409.0000
 1.0    1      H     O           7557.0000      2385.0000
 1.0    1      H     OH          7557.0000      2385.0000
 1.0    3      H     S         265720.0000     35429.0000
 1.0    3      H     SH        265720.0000     35429.0000
 1.0    1      HO    NB          7557.0000      2385.0000
 1.0    1      HO    NC          7557.0000      2385.0000
 1.0    1      HO    O2          4019.0000      1409.0000
 1.0    1      HO    O           7557.0000      2385.0000
 1.0    1      HO    OH          7557.0000      2385.0000
 1.0    3      HO    S         265720.0000     35429.0000
 1.0    3      HO    SH        265720.0000     35429.0000
 1.0    1      H2    NB          4019.0000      1409.0000
 1.0    1      H2    NC          4019.0000      1409.0000
 1.0    1      H2    O2          4019.0000      1409.0000
 1.0    1      H2    O          10238.0000      3071.0000
 1.0    1      H2    OH          4019.0000      1409.0000
 1.0    3      H2    S         265720.0000     35429.0000
 1.0    3      H2    SH        265720.0000     35429.0000
 1.0    1      H3    NB          4019.0000      1409.0000
 1.0    1      H3    NC          4019.0000      1409.0000
 1.0    1      H3    O2          4019.0000      1409.0000
 1.0    1      H3    O           7557.0000      2385.0000
 1.0    1      H3    OH          7557.0000      2385.0000
 1.0    3      H3    S         265720.0000     35429.0000
 1.0    3      H3    SH        265720.0000     35429.0000
 1.0    1      HS    NB         14184.0000      3082.0000
 1.0    1      HS    NC         14184.0000      3082.0000
 1.0    1      HS    O2         14184.0000      3082.0000
 1.0    1      HS    O          14184.0000      3082.0000
 1.0    1      HS    OH         14184.0000      3082.0000
 1.0    3      HS    S         265720.0000     35429.0000
 1.0    3      HS    SH        265720.0000     35429.0000
#bond_increments     amber
!Ver  Ref      I      J        DeltaIJ      DeltaJI
!---- ---     ----   ----      -------      -------
 1.1    5      CM     CM        0.000        0.000
 1.1    5      CA     CA        0.000        0.000
 1.1    5      CB     CB        0.000        0.000
```

1.1	5	C5	C6	0.000	0.000
1.1	5	CT	CT	0.000	0.000
1.1	5	HT	CT	0.066	-0.066
1.1	5	H	NT	0.133	-0.133
1.1	5	NT	CT	-0.189	0.189
1.1	5	CA	OH	0.334	-0.334
1.1	5	CT	OS	0.237	-0.237
1.1	5	HC	CT	0.066	-0.066
1.1	6	CS	CS	0.000	0.000
1.1	6	AC	CS	0.000	0.000
1.1	6	BC	CS	0.000	0.000
1.1	6	CS	CT	0.000	0.000
1.1	6	CS	OS	0.200	-0.200
1.1	5	N*	CS	-0.183	0.183
1.1	6	OT	HY	-0.400	0.400
1.1	6	OA	HY	-0.400	0.400
1.1	6	OB	HY	-0.400	0.400
1.1	6	CS	HT	-0.100	0.100
1.1	5	AC	AH	-0.100	0.100
1.1	6	BC	BH	-0.100	0.100
1.1	6	AC	HT	-0.100	0.100
1.1	6	BC	HT	-0.100	0.100
1.1	6	AC	OA	0.250	-0.250
1.1	6	BC	OB	0.250	-0.250
1.1	6	CS	OA	0.250	-0.250
1.1	6	CS	OB	0.250	-0.250
1.1	6	CS	OT	0.250	-0.250
1.1	6	CS	OE	0.200	-0.200
1.1	6	AC	OE	0.200	-0.200
1.1	5	BC	OE	0.200	-0.200
1.1	6	OW	HW	-0.380	0.380
1.1	5	N*	CT	-0.183	0.183
1.1	5	P	OS	0.254	-0.254
1.1	5	CB	N*	0.130	-0.130
1.1	5	CK	N*	-0.253	0.253
1.1	5	NC	CB	-0.335	0.335
1.1	5	NB	CB	0.020	-0.020
1.1	5	CB	CA	0.000	-0.000
1.1	5	CK	NB	0.566	-0.566

1.1	5	CK	HC	-0.051	0.051
1.1	5	N2	CA	-0.162	0.162
1.1	5	NC	CA	-0.430	0.430
1.1	5	H2	N2	0.318	-0.318
1.1	5	CQ	NC	0.341	-0.341
1.1	5	CQ	HC	0.005	-0.005
1.1	5	O2	P	-0.913	0.413
1.1	5	C	N*	-0.044	0.044
1.1	5	CM	N*	0.137	-0.137
1.1	5	NA	C	-0.255	0.255
1.1	5	O	C	-0.492	0.492
1.1	5	NA	H	-0.282	0.282
1.1	5	CM	C	-0.150	0.150
1.1	5	CM	CT	0.055	-0.055
1.1	5	CM	HC	-0.101	0.101
1.1	5	H2	CT	0.119	-0.119
1.1	5	C	NC	0.424	-0.424
1.1	5	CM	CA	-0.409	0.409
1.1	5	N2	HC	-0.037	0.037
1.1	5	OH	CT	-0.263	0.263
1.1	5	HO	OH	0.303	-0.303
1.1	5	C	CB	-0.005	0.005
1.1	5	NA	CA	-0.215	0.215
1.1	5	CT	N	0.171	-0.171
1.1	5	H	N	0.274	-0.274
1.1	5	C	CT	0.095	-0.095
1.1	5	C	N	0.139	-0.139
1.1	5	N2	CT	0.044	-0.044
1.1	5	H3	N2	0.551	-0.351
1.1	5	O2	C	-0.792	0.292
1.1	5	S	CT	-0.023	0.023
1.1	5	LP	S	-0.403	0.403
1.1	5	SH	CT	-0.033	0.033
1.1	5	HS	SH	0.127	-0.127
1.1	5	SH	LP	0.489	-0.489
1.1	5	CC	CT	0.007	-0.007
1.1	5	NB	CC	-0.256	0.256
1.1	5	CW	CC	0.018	-0.018
1.1	5	CR	NB	0.251	-0.251

```
 1.1    5      NA     CR        -0.066        0.066
 1.1    5      CR     HC        -0.067        0.067
 1.1    5      CW     NA        -0.057        0.057
 1.1    5      CW     HC        -0.099        0.099
 1.1    5      NA     CC        -0.020        0.020
 1.1    5      NA     PS         0.423       -0.423
 1.1    5      CV     CC         0.035       -0.035
 1.1    5      CV     NB         0.227       -0.227
 1.1    5      CV     HC        -0.042        0.042
 1.1    5      N3     CT         0.905        0.095
 1.1    5      N3     H3        -0.326        0.326
 1.1    5      CA     CT        -0.033        0.033
 1.1    5      CA     HC        -0.101        0.101
 1.1    5      C*     CT         0.005       -0.005
 1.1    5      C*     CW        -0.192        0.192
 1.1    5      CB     C*        -0.045        0.045
 1.1    5      CN     NA         0.176       -0.176
 1.1    5      CN     CA         0.074       -0.074
 1.1    5      CB     CN         0.104       -0.104
 1.1    5      CA     C         -0.181        0.181
 1.1    5      OH     C         -0.081        0.081
#reference 1
creation of file
#reference 2
Lone pair lp had incorrect mass of 0.001097.
Angle CT-C-O2 was by error included twice.
Torsion OH-C2-C2-OH was written as two separate lines.
Hence only one of the energy terms was included.
@Author Jon Hurley
@Date 13-December-90
#reference 3
parameter set modified with the addtional parameters
from kollman's parm89a rev a force field file
note that the HW...OW hydrogen bond parameters and
the HW van der waals parameters are not included in
the files since they are equal to zero in parm89a.
@Author tom thacher
@Date 11-March-92
#refer nce 4
```

```
homans' carbohydrate potential
@Author Tom Thacher
@Date 7-July-1992
#reference 5
bond increments
@Author Tom Thacher
@Date 7-July-1992
#end

****************************************************
****************************************************
                       END OF LISTING
*******************************************************
******************************************************

****************************************************
           DATA FILE FOR H BOND FORCES - HBOND.DAT
*****************************************************

47 !data items
!BIOSYM forcefield          2
!version amber.frc  1.0  19-Oct-90
!version amber.frc  1.1   8-Aug-92
!define amber
! This is the new format version of the amber forcefield
!hbond_definition       amber
!1.0    1   distance      2.5000
!1.0    1   angle        90.0000
!1.0    1   donors        H    HO   H2   H3   HS
!1.0    1   acceptors     NB   NC   O2   O    OH   S    SH
!hydrogen_bond(10-12)     amber
! E = Aij/r^12 - Bij/r^10
!Ver  Ref      I    J          A               B
!---- ---     ---- ----    -----------    -----------
 1.0   3      H    OS         7557.0000      2385.0000
 1.0   3      H    OW         7557.0000      2385.0000
 1.0   3      H2   OS         7557.0000      2385.0000
 1.0   3      H2   OW         7557.0000      2385.0000
 1.0   3      HW   NB         7557.0000      2385.0000
```

1.0	3	HW	NC	10238.0000	3071.0000
1.0	3	HW	O	7557.0000	2385.0000
1.0	3	HW	O2	4019.0000	1409.0000
1.0	3	HW	OH	7557.0000	2385.0000
1.0	3	HW	OS	7557.0000	2385.0000
1.0	3	HW	S	265720.0000	35429.0000
1.0	3	HW	SH	265720.0000	35429.0000
1.0	1	H	NB	7557.0000	2385.0000
1.0	1	H	NC	10238.0000	3071.0000
1.0	1	H	O2	4019.0000	1409.0000
1.0	1	H	O	7557.0000	2385.0000
1.0	1	H	OH	7557.0000	2385.0000
1.0	3	H	S	265720.0000	35429.0000
1.0	3	H	SH	265720.0000	35429.0000
1.0	1	HO	NB	7557.0000	2385.0000
1.0	1	HO	NC	7557.0000	2385.0000
1.0	1	HO	O2	4019.0000	1409.0000
1.0	1	HO	O	7557.0000	2385.0000
1.0	1	HO	OH	7557.0000	2385.0000
1.0	3	HO	S	265720.0000	35429.0000
1.0	3	HO	SH	265720.0000	35429.0000
1.0	1	H2	NB	4019.0000	1409.0000
1.0	1	H2	NC	4019.0000	1409.0000
1.0	1	H2	O2	4019.0000	1409.0000
1.0	1	H2	O	10238.0000	3071.0000
1.0	1	H2	OH	4019.0000	1409.0000
1.0	3	H2	S	265720.0000	35429.0000
1.0	3	H2	SH	265720.0000	35429.0000
1.0	1	H3	NB	4019.0000	1409.0000
1.0	1	H3	NC	4019.0000	1409.0000
1.0	1	H3	O2	4019.0000	1409.0000
1.0	1	H3	O	7557.0000	2385.0000
1.0	1	H3	OH	7557.0000	2385.0000
1.0	3	H3	S	265720.0000	35429.0000
1.0	3	H3	SH	265720.0000	35429.0000
1.0	1	HS	NB	14184.0000	3082.0000
1.0	1	HS	NC	14184.0000	3082.0000
1.0	1	HS	O2	14184.0000	3082.0000
1.0	1	HS	O	14184.0000	3082.0000

```
 1.0   1      HS   OH       14184.0000     3082.0000
 1.0   3      HS   S       265720.0000    35429.0000
 1.0   3      HS   SH      265720.0000    35429.0000

**********************************************************
      DATA FILE FOR LENNARD JONES FORCES - LJ_PARAM.DAT
************************************************************

74 !total atoms
!BIOSYM forcefield          2
!version amber.frc  1.0  19-Oct-90
!version amber.frc  1.1   8-Aug-92
!define amber
! This is the new format version of the amber forcefield
!nonbond(12-6) amber
!type r-eps
!combination arithmetic
! E = EPSij * { (Rij*/Rij)^12 - 2(Rij*/Rij)^6 }
! where  EPSij = sqrt( EPSi * EPSj)
!          Rij* = (Ri* + Rj*)/2
!Ver  Ref      I            Ri*          EPSi
!---- ---     ----      -----------   -----------
 1.0   3      IM           5.0000        0.10000
 1.0   3      CU           2.4000        0.05000
 1.0   3      I            4.8000        0.40000
 1.0   3      OW           3.5360        0.15200
 1.0   3      MG           2.3400        0.10000
 1.0   3      C0           3.2000        0.10000
 1.0   3      QC           6.8000        0.00008
 1.0   3      QK           5.3200        0.00033
 1.0   3      QL           2.2800        0.01800
 1.0   3      QN           3.7400        0.00280
 1.0   3      QR           5.9200        0.00017
 1.0   1      C            3.7000        0.12000
 1.0   1      C*           3.7000        0.12000
 1.0   1      C2           3.8400        0.12000
 1.0   1      C3           4.0000        0.15000
 1.0   1      CA           3.7000        0.12000
 1.0   1      CB           3.7000        0.12000
```

1.0	1	CC	3.7000	0.12000
1.0	1	CD	3.7000	0.12000
1.0	1	CE	3.7000	0.12000
1.0	1	CF	3.7000	0.12000
1.0	1	CG	3.7000	0.12000
1.0	1	CH	3.7000	0.09000
1.0	1	CI	3.7000	0.12000
1.0	1	CJ	3.7000	0.12000
1.0	1	CK	3.7000	0.12000
1.0	1	CM	3.7000	0.12000
1.0	1	CN	3.7000	0.12000
1.0	1	CP	3.7000	0.12000
1.0	1	CQ	3.7000	0.12000
1.0	1	CR	3.7000	0.12000
1.0	1	CT	3.6000	0.06000
1.0	1	CV	3.7000	0.12000
1.0	1	CW	3.7000	0.12000
1.0	1	H	2.0000	0.02000
1.0	1	H2	2.0000	0.02000
1.0	1	H3	2.0000	0.02000
1.0	1	HC	3.0800	0.01000
1.0	1	HO	2.0000	0.02000
1.0	1	HS	2.0000	0.02000
1.0	1	LP	2.4000	0.01600
1.0	1	N	3.5000	0.16000
1.0	1	N*	3.5000	0.16000
1.0	1	N2	3.5000	0.16000
1.0	1	N3	3.7000	0.08000
1.0	1	NA	3.5000	0.16000
1.0	1	NB	3.5000	0.16000
1.0	1	NC	3.5000	0.16000
1.0	1	NP	3.5000	0.16000
1.0	1	NT	3.7000	0.12000
1.0	1	O	3.2000	0.20000
1.0	1	O2	3.2000	0.20000
1.0	1	OH	3.3000	0.15000
1.0	1	OS	3.3000	0.15000
1.0	1	P	4.2000	0.20000
1.0	1	S	4.0000	0.20000

```
 1.0   1     SH            4.0000        0.20000
 1.1   4     CS            3.6000        0.09030
 1.1   4     AC            3.6000        0.09030
 1.1   4     BC            3.6000        0.09030
 1.1   4     C             3.7000        0.12000
 1.1   4     H             2.0000        0.02000
 1.1   4     HY            1.6000        0.04980
 1.1   4     HT            2.9360        0.00450
 1.1   4     HO            2.0000        0.02000
 1.1   4     AH            2.9360        0.00450
 1.1   4     BH            2.9360        0.00450
 1.1   4     OT            3.2000        0.15910
 1.1   4     OA            3.2000        0.15910
 1.1   4     OB            3.2000        0.15910
 1.1   4     OE            3.2000        0.15910
 1.1   4     OH            3.3000        0.15000
 1.1   4     O             3.2000        0.20000
 1.1   4     N             3.5000        0.16000

*******************************************************
          DATA FILE FOR TORSION FORCES - TORSION.DAT
*******************************************************

179 ! total entries in this data file
!BIOSYM forcefield          2
!version amber.frc  1.0  19-Oct-90
!version amber.frc  1.1   8-Aug-92
!define amber
! This is the new format version of the amber forcefield
!torsion_3      amber
! E = SUM(n=1,3) { V(n) * [ 1 + cos(n*Phi - Phi0(n)) ] }
!Ver  Ref      I     J     K     L          V1      Phi0
V2      Phi0         V3      Phi0
!---- ---      ----  ----  ----  ----       -------  ------
------- ------     ------- -----
 1.0   1      O      C     C2    N          0.0000     0.0
0.0000     0.0      0.2000  180.0
 1.0   1      O      C     CH    C2         0.0000     0.0
0.0000     0.0      0.1000  180.0
```

```
 1.0    1      O      C      CH     N              0.0000     0.0
0.0000    0.0      0.1000  180.0
 1.0    1      O      C      CH     CH             0.0000     0.0
0.0000    0.0      0.1000  180.0
 1.0    1      OS     C2     C2     OH             0.0000     0.0
0.5000    0.0      2.0000    0.0
 1.0    2      OH     C2     C2     OH             0.0000     0.0
0.5000    0.0      2.0000    0.0
 1.0    1      OS     C2     C2     OS             0.0000     0.0
0.5000    0.0      2.0000    0.0
 1.0    1      OS     C2     CH     OS             0.0000     0.0
0.5000    0.0      1.0000    0.0
 1.0    1      OS     C2     CH     OH             0.0000     0.0
0.5000    0.0      1.0000    0.0
 1.0    1      OH     C2     CH     OH             0.0000     0.0
0.5000    0.0      1.0000    0.0
 1.0    1      C2     C2     S      LP             0.0000     0.0
0.0000    0.0      0.0000    0.0
 1.0    1      CH     C2     SH     LP             0.0000     0.0
0.0000    0.0      0.0000    0.0
 1.0    1      OS     CH     C2     OH             0.0000     0.0
0.5000    0.0      1.0000    0.0
 1.0    1      OH     CH     CH     OH             0.0000     0.0
0.5000    0.0      0.5000    0.0
 1.0    1      OS     CH     CH     OH             0.0000     0.0
0.5000    0.0      0.5000    0.0
 1.0    1      OS     CH     CH     OS             0.0000     0.0
0.5000    0.0      0.5000    0.0
 1.0    1      HC     CM     CM     CT             0.0000     0.0
1.7100  180.0      0.0000    0.0
 1.0    1      C      CM     CM     HC             0.0000     0.0
6.5900  180.0      0.0000    0.0
 1.0    1      N*     CM     CM     CT             0.0000     0.0
6.5900  180.0      0.0000    0.0
 1.0    1      CA     CM     CM     HC             0.0000     0.0
6.5900  180.0      0.0000    0.0
 1.0    1      N*     CM     CM     CA             0.0000     0.0
9.5100  180.0      0.0000    0.0
 1.0    1      HC     CM     CM     HC             0.0000     0.0
```

```
1.7100  180.0       0.0000    0.0
 1.0     1      N*     CM     CM     C             0.0000    0.0
9.5100  180.0       0.0000    0.0
 1.0     1      N*     CM     CM     HC            0.0000    0.0
6.5900  180.0       0.0000    0.0
 1.0     1      N      CT     C      O             0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0     1      HC     CT     C      O             0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0     1      CT     CT     C      O             0.0000    0.0
0.0000    0.0       0.0670  180.0
 1.0     1      CT     OS     CT     CT            0.0000    0.0
0.2000  180.0       0.3830    0.0
 1.0     1      OS     CT     CT     OS            0.0000    0.0
0.5000    0.0       0.1440    0.0
 1.0     1      OS     CT     CT     OH            0.0000    0.0
0.5000    0.0       0.1440    0.0
 1.0     1      OH     CT     CT     OH            0.0000    0.0
0.5000    0.0       0.1440    0.0
 1.0     1      H      N      C      O             0.6500    0.0
2.5000  180.0       0.0000    0.0
 1.0     1      C2     OS     C2     C3            0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0     1      C2     OS     C2     C2            0.0000    0.0
0.1000    0.0       1.4500    0.0
 1.0     1      C3     OS     C2     C3            0.0000    0.0
0.1000    0.0       1.4500    0.0
 1.0     1      CH     OS     CH     C2            0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0     1      CH     OS     CH     CH            0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0     1      C2     OS     CH     C2            0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0     1      C3     OS     CH     C3            0.0000    0.0
0.1000    0.0       0.7250    0.0
 1.0     1      CH     OS     CH     N*            0.0000    0.0
0.0000    0.0       0.7250    0.0
 1.0     1      C2     OS     CH     C3            0.0000    0.0
0.1000    0.0       0.7250    0.0
```

```
 1.0    1       OH      P       OS      C3              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OS      P       OS      C2              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OH      P       OS      C2              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OS      P       OS      CT              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OS      P       OS      CH              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OS      P       OS      C3              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OH      P       OS      CH              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       OH      P       OS      CT              0.0000     0.0
0.7500     0.0        0.2500     0.0
 1.0    1       LP      S       S       LP              0.0000     0.0
0.0000     0.0        0.0000     0.0
 1.0    1       LP      S       S       C2              0.0000     0.0
0.0000     0.0        0.0000     0.0
 1.0    1       C2      S       S       C2              0.0000     0.0
3.5000     0.0        0.6000     0.0
 1.0    1       CT      S       S       CT              0.0000     0.0
3.5000     0.0        0.6000     0.0
 1.0    1       LP      S       S       CT              0.0000     0.0
0.0000     0.0        0.0000     0.0
 1.1    4       OE      AC      OA      CS              2.1500   300.0
0.0000     0.0        0.0000     0.0
 1.1    4       AH      AC      OA      CS              0.0000     0.0
1.7500    60.0        0.0000     0.0
 1.1    4       CS      AC      OA      CS              0.0000     0.0
0.0000     0.0        0.8500     0.0
 1.1    4       OE      AC      OA      HY              2.1500   300.0
0.0000     0.0        0.0000     0.0
 1.1    4       AH      AC      OA      HY              0.0000     0.0
1.7500    60.0        0.0000     0.0
 1.1    4       CS      AC      OA      HY              0.0000     0.0
0.0000     0.0        0.8500     0.0
 1.1    4       OE      BC      OB      CS             -1.0500     0.0
```

```
0.0000    0.0       0.0000    0.0
 1.1    4       BH      BC      OB      CS            0.0000      0.0
1.2500  240.0       0.0000    0.0
 1.1    4       CS      BC      OB      CS            0.0000      0.0
0.0000    0.0       1.4000    0.0
 1.1    4       OE      BC      OB      HY           -1.0500      0.0
0.0000    0.0       0.0000    0.0
 1.1    4       BH      BC      OB      HY            0.0000      0.0
1.2500  240.0       0.0000    0.0
 1.1    4       CS      BC      OB      HY            0.0000      0.0
0.0000    0.0       1.4000    0.0
 1.1    4       HT      AC      OA      CS            0.0000      0.0
0.0000    0.0       0.8500    0.0
 1.1    4       HT      BC      OB      CS            0.0000      0.0
0.0000    0.0       1.4000    0.0
 1.1    4       H       N       C       O             0.6500      0.0
2.5000  180.0       0.0000    0.0
 1.1    4       HT      CS      C       O             0.0000      0.0
0.0000    0.0       0.0670  180.0
 1.0    3       *       CB      CD      *             0.0000      0.0
5.3000  180.0       0.0000    0.0
 1.0    1       *       C       C2      *             0.0000      0.0
0.0000    0.0       0.0000  180.0
 1.0    1       *       C       CA      *             0.0000      0.0
5.3000  180.0       0.0000    0.0
 1.0    1       *       C       CB      *             0.0000      0.0
4.4000  180.0       0.0000    0.0
 1.0    1       *       C       CD      *             0.0000      0.0
5.3000  180.0       0.0000    0.0
 1.0    1       *       C       CH      *             0.0000      0.0
0.0000    0.0       0.0000    0.0
 1.0    1       *       C       CJ      *             0.0000      0.0
3.1000  180.0       0.0000    0.0
 1.0    1       *       C       CM      *             0.0000      0.0
3.1000  180.0       0.0000    0.0
 1.0    1       *       C       CT      *             0.0000      0.0
0.0000    0.0       0.0000    0.0
 1.0    1       *       C       N       *             0.0000      0.0
10.0000  180.0      0.0000    0.0
```

```
 1.0    1      *      C      N*     *              0.0000     0.0
5.8000  180.0       0.0000    0.0
 1.0    1      *      C      NA     *              0.0000     0.0
5.4000  180.0       0.0000    0.0
 1.0    1      *      C      NC     *              0.0000     0.0
8.0000  180.0       0.0000    0.0
 1.0    1      *      C      OH     *              0.0000     0.0
1.8000  180.0       0.0000    0.0
 1.0    1      *      C*     C2     *              0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C*     CB     *              0.0000     0.0
4.8000  180.0       0.0000    0.0
 1.0    1      *      C*     CG     *              0.0000     0.0
23.6000  180.0       0.0000    0.0
 1.0    1      *      C*     CT     *              0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C*     CW     *              0.0000     0.0
23.6000  180.0       0.0000    0.0
 1.0    1      *      C2     C2     *              0.0000     0.0
0.0000    0.0       2.0000    0.0
 1.0    1      *      C2     CA     *              0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2     CC     *              0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2     CH     *              0.0000     0.0
0.0000    0.0       2.0000    0.0
 1.0    1      *      C2     N      *              0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2     N2     *              0.0000     0.0
0.0000    0.0       0.0000    0.0
 1.0    1      *      C2     N3     *              0.0000     0.0
0.0000    0.0       1.4000    0.0
 1.0    1      *      C2     NT     *              0.0000     0.0
0.0000    0.0       1.0000    0.0
 1.0    1      *      C2     OH     *              0.0000     0.0
0.0000    0.0       0.5000    0.0
 1.0    1      *      C2     OS     *              0.0000     0.0
0.0000    0.0       1.4500    0.0
 1.0    1      *      C2     S      *              0.0000     0.0
```

```
0.0000     0.0       1.0000     0.0
 1.0     1       *      C2      SH      *          0.0000     0.0
0.0000     0.0       0.7500     0.0
 1.0     1       *      CA      CA      *          0.0000     0.0
5.3000   180.0       0.0000     0.0
 1.0     1       *      CA      CB      *          0.0000     0.0
10.2000   180.0      0.0000     0.0
 1.0     1       *      CA      CD      *          0.0000     0.0
5.3000   180.0       0.0000     0.0
 1.0     1       *      CA      CJ      *          0.0000     0.0
3.7000   180.0       0.0000     0.0
 1.0     1       *      CA      CM      *          0.0000     0.0
3.7000   180.0       0.0000     0.0
 1.0     1       *      CA      CN      *          0.0000     0.0
10.6000   180.0      0.0000     0.0
 1.0     1       *      CA      CT      *          0.0000     0.0
0.0000     0.0       0.0000     0.0
 1.0     1       *      CA      N2      *          0.0000     0.0
6.8000   180.0       0.0000     0.0
 1.0     1       *      CA      NA      *          0.0000     0.0
6.0000   180.0       0.0000     0.0
 1.0     1       *      CA      NC      *          0.0000     0.0
9.6000   180.0       0.0000     0.0
 1.0     1       *      CB      CB      *          0.0000     0.0
16.3000   180.0      0.0000     0.0
 1.0     1       *      CB      CN      *          0.0000     0.0
20.0000   180.0      0.0000     0.0
 1.0     1       *      CB      N*      *          0.0000     0.0
6.6000   180.0       0.0000     0.0
 1.0     1       *      CB      NB      *          0.0000     0.0
5.1000   180.0       0.0000     0.0
 1.0     3       *      CB      NC      *          0.0000     0.0
8.3000   180.0       0.0000     0.0
 1.0     1       *      CC      CF      *          0.0000     0.0
14.3000   180.0      0.0000     0.0
 1.0     1       *      CC      CG      *          0.0000     0.0
15.9000   180.0      0.0000     0.0
 1.0     1       *      CC      CT      *          0.0000     0.0
0.0000     0.0     , 0.0000     0.0
```

```
 1.0    1       *        CC      CV      *             0.0000      0.0
14.3000  180.0       0.0000     0.0
 1.0    1       *        CC      CW      *             0.0000      0.0
15.9000  180.0       0.0000     0.0
 1.0    1       *        CC      NA      *             0.0000      0.0
5.6000  180.0       0.0000     0.0
 1.0    1       *        CC      NB      *             0.0000      0.0
4.8000  180.0       0.0000     0.0
 1.0    1       *        CD      CD      *             0.0000      0.0
5.3000  180.0       0.0000     0.0
 1.0    1       *        CD      CN      *             0.0000      0.0
5.3000  180.0       0.0000     0.0
 1.0    1       *        CE      N*      *             0.0000      0.0
6.7000  180.0       0.0000     0.0
 1.0    1       *        CE      NB      *             0.0000      0.0
20.0000  180.0       0.0000     0.0
 1.0    1       *        CF      NB      *             0.0000      0.0
4.8000  180.0       0.0000     0.0
 1.0    1       *        CG      NA      *             0.0000      0.0
6.0000  180.0       0.0000     0.0
 1.0    1       *        CH      CH      *             0.0000      0.0
0.0000     0.0       2.0000     0.0
 1.0    1       *        CH      N       *             0.0000      0.0
0.0000     0.0       0.0000     0.0
 1.0    1       *        CH      N*      *             0.0000      0.0
0.0000     0.0       0.0000     0.0
 1.0    1       *        CH      NT      *             0.0000      0.0
0.0000     0.0       1.0000     0.0
 1.0    1       *        CH      OH      *             0.0000      0.0
0.0000     0.0       0.5000     0.0
 1.0    1       *        CH      OS      *             0.0000      0.0
0.0000     0.0       1.4500     0.0
 1.0    1       *        CI      NC      *             0.0000      0.0
13.5000  180.0       0.0000     0.0
 1.0    1       *        CJ      CJ      *             0.0000      0.0
24.4000  180.0       0.0000     0.0
 1.0    1       *        CJ      CM      *             0.0000      0.0
24.4000  180.0       0.0000     0.0
 1.0    1       *        CJ      N*      *             0.0000      0.0
```

```
7.4000   180.0       0.0000     0.0
 1.0    1       *     CK     N*     *              0.0000     0.0
6.7000   180.0       0.0000     0.0
 1.0    1       *     CK     NB     *              0.0000     0.0
20.0000   180.0      0.0000     0.0
 1.0    1       *     CM     CM     *              0.0000     0.0
24.4000   180.0      0.0000     0.0
 1.0    1       *     CM     CT     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
 1.0    1       *     CM     N*     *              0.0000     0.0
7.4000   180.0       0.0000     0.0
 1.0    1       *     CN     NA     *              0.0000     0.0
12.2000   180.0      0.0000     0.0
 1.0    1       *     CP     NA     *              0.0000     0.0
9.3000   180.0       0.0000     0.0
 1.0    1       *     CP     NB     *              0.0000     0.0
10.0000   180.0      0.0000     0.0
 1.0    1       *     CQ     NC     *              0.0000     0.0
13.5000   180.0      0.0000     0.0
 1.0    1       *     CR     NA     *              0.0000     0.0
9.3000   180.0       0.0000     0.0
 1.0    1       *     CR     NB     *              0.0000     0.0
10.0000   180.0      0.0000     0.0
 1.0    1       *     CT     CT     *              0.0000     0.0
0.0000     0.0       1.3000     0.0
 1.0    1       *     CT     N      *              0.0000     0.0
0.0000     0.0       0.0000     0.0
 1.0    1       *     CT     N*     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
 1.0    1       *     CT     N2     *              0.0000     0.0
0.0000     0.0       0.0000     0.0
 1.0    1       *     CT     N3     *              0.0000     0.0
0.0000     0.0       1.4000     0.0
 1.0    1       *     CT     OH     *              0.0000     0.0
0.0000     0.0       0.5000     0.0
 1.0    1       *     CT     OS     *              0.0000     0.0
0.0000     0.0       1.1500     0.0
 1.0    1       *     CT     S      *              0.0000     0.0
0.0000     0.0       1.0000     0.0
```

```
 1.0    1      *     CT    SH    *          0.0000    0.0
0.0000    0.0    0.7500    0.0
 1.0    1      *     CV    NB    *          0.0000    0.0
4.8000  180.0    0.0000    0.0
 1.0    1      *     CW    NA    *          0.0000    0.0
6.0000  180.0    0.0000    0.0
 1.0    1      *     OH    P     *          0.0000    0.0
0.0000    0.0    0.7500    0.0
 1.0    1      *     OS    P     *          0.0000    0.0
0.0000    0.0    0.7500    0.0
 1.1    4      *     CS    CS    *          0.0000    0.0
0.0000    0.0    1.0210    0.0
 1.1    4      *     CS    CT    *          0.0000    0.0
0.0000    0.0    1.0210    0.0
 1.1    4      *     AC    CS    *          0.0000    0.0
0.0000    0.0    1.0210    0.0
 1.1    4      *     BC    CS    *          0.0000    0.0
0.0000    0.0    1.0210    0.0
 1.1    4      *     CS    OT    *          0.0000    0.0
0.0000    0.0    0.4430    0.0
 1.1    4      *     CS    OE    *          0.0000    0.0
0.0000    0.0    0.9280    0.0
 1.1    4      *     AC    OE    *          0.0000    0.0
0.0000    0.0    0.9280    0.0
 1.1    4      *     BC    OE    *          0.0000    0.0
0.0000    0.0    0.9280    0.0
 1.1    4      *     AC    OA    *          0.0000    0.0
0.0000    0.0    0.0000    0.0
 1.1    4      *     BC    OB    *          0.0000    0.0
0.0000    0.0    0.0000    0.0
 1.1    4      *     CS    OA    *          0.0000    0.0
0.0000    0.0    0.0000    0.0
 1.1    4      *     CS    OB    *          0.0000    0.0
0.0000    0.0    0.0000    0.0
 1.1    4      *     CS    N     *          0.0000    0.0
0.0000    0.0    0.0000    0.0
 1.1    4      *     C     N     *          0.0000    0.0
10.0000  180.0    0.0000    0.0
 1.1    4      *     C     CS    *          0.0000    0.0
```

```
0.0000    0.0      0.0000    0.0
 1.0    1      *      CT     NT     *          0.0000     0.0
0.0000    0.0      1.8000    0.0

********************************************************

                  DATA FILE - CX6C.CAR

********************************************************

!BIOSYM archive 3
PBC=OFF
!DATE Thu Mar  2 10:02:29 1995
SG         0.051616628    8.775964550    2.653307337 CYSn 1
S        S   0.824
LG1       -0.116704460    8.906803991    3.732450018 CYSn 1
LP       L  -0.405
LG2       -0.816371929    8.216369655    2.274560255 CYSn 1
LP       L  -0.405
CB         1.625257994    7.970290997    2.280061368 CYSn 1
CT       C  -0.098
HB1        1.743097230    7.117856362    2.972980432 CYSn 1
HC       H   0.050
HB2        2.457560406    8.667686711    2.506611212 CYSn 1
HC       H   0.050
CA         1.664891168    7.503978115    0.811322158 CYSn 1
CT       C   0.035
HA         2.715618613    7.453348875    0.469159517 CYSn 1
HC       H   0.032
N          0.954382540    8.512673633    0.003030230 CYSn 1
NT       N  -0.463
C          1.063568189    6.132700222    0.616111991 CYSn 1
C        C   0.616
O          0.248707622    5.654726837    1.414398016 CYSn 1
O        O  -0.504
N          1.449902196    5.479885680   -0.464156147 GLY  2
N        N  -0.463
HN         2.157106102    5.992384244   -1.099457509 GLY  2
H        H   0.252
CA         0.868490592    4.154014497   -0.652902307 GLY  2
CT       C   0.035
```

```
HA1       1.550908149     3.403064022     -0.212395307 GLY  2
HC       H   0.032
HA2      -0.097660558     4.132736815     -0.116611463 GLY  2
HC       H   0.032
C         0.730531165     3.827591429     -2.120728786 GLY  2
C        C   0.616
O         1.559375145     4.206208097     -2.957020570 GLY  2
O        O  -0.504
N        -0.320742949     3.103195380     -2.456098946 GLY  3
N        N  -0.463
HN       -0.976177839     2.817016114     -1.646836012 GLY  3
H        H   0.252
CA       -0.454134161     2.787581074     -3.875321662 GLY  3
CT       C   0.035
HA1      -0.907422830     1.783240810     -3.972773051 GLY  3
HC       H   0.032
HA2      -1.127648566     3.540414569     -4.323795441 GLY  3
HC       H   0.032
C         0.896974016     2.736484179     -4.547627543 GLY  3
C        C   0.616
O         1.315189212     1.712629073     -5.101282348 GLY  3
O        O  -0.504
N         1.599575272     3.853622667     -4.520184621 GLY  4
N        N  -0.463
HN        1.137216234     4.691535216     -4.019658253 GLY  4
H        H   0.252
CA        2.905944550     3.804217731     -5.170228610 GLY  4
CT       C   0.035
HA1       3.056204584     2.789614618     -5.584558431 GLY  4
HC       H   0.032
HA2       2.897891721     4.540755026     -5.994216851 GLY  4
HC       H   0.032
C         4.014980067     4.050747291     -4.175561433 GLY  4
C        C   0.616
O         4.978871195     4.780583329     -4.436272241 GLY  4
O        O  -0.504
N         3.887759074     3.450944950     -3.006608050 GLY  5
N        N  -0.463
HN        3.003276191     2.844372268     -2.879487738 GLY  5
```

```
H          H    0.252
CA          4.960071382     3.689311240    -2.044877031 GLY  5
CT         C    0.035
HA1         5.709592998     2.881830301    -2.144167698 GLY  5
HC         H    0.032
HA2         5.427393718     4.658369322    -2.297948016 GLY  5
HC         H    0.032
C           4.437174470     3.643619035    -0.629041435 GLY  5
C          C    0.616
O           3.798322352     2.676595378    -0.197242766 GLY  5
O          O   -0.504
N           4.713663113     4.691871185     0.124033264 GLY  6
N          N   -0.463
HN          5.286002166     5.476492875    -0.348403798 GLY  6
H          H    0.252
CA          4.208080753     4.647691975     1.492986659 GLY  6
CT         C    0.035
HA1         3.303800182     4.010943092     1.515218779 GLY  6
HC         H    0.032
HA2         4.993057374     4.194323221     2.125265975 GLY  6
HC         H    0.032
C           3.799265981     6.023038258     1.963510280 GLY  6
C          C    0.616
O           4.006824522     7.036283245     1.285298717 GLY  6
O          O   -0.504
N           3.195690211     6.077750863     3.136158080 GLY  7
N          N   -0.463
HN          3.055107813     5.133307510     3.640799839 GLY  7
H          H    0.252
CA          2.800412417     7.407555656     3.591101372 GLY  7
CT         C    0.035
HA1         1.946687677     7.303619509     4.286815466 GLY  7
HC         H    0.032
HA2         3.660862081     7.847316876     4.127520148 GLY  7
HC         H    0.032
C           2.334578164     8.258959996     2.434291753 GLY  7
C          C    0.616
O           2.337411236     9.494643783     2.487154063 GLY  7
O          O   -0.504
```

```
N         1.936206121    7.605756209    1.358640986 CYSN 8
N       N  -0.463
HN        1.983632457    6.528240768    1.414418956 CYSN 8
H       H   0.252
CA        1.485796919    8.428968216    0.240136508 CYSN 8
CT      C   0.035
HA        0.399931102    8.271042216    0.100059529 CYSN 8
HC      H   0.032
C         2.167493478    8.018162291   -1.043072620 CYSN 8
C       C   0.616
CB        1.746659419    9.902481747    0.610166221 CYSN 8
CT      C  -0.098
HB1       2.709270705   10.016688002    1.140264476 CYSN 8
HC      H   0.050
HB2       1.816139488   10.541353385   -0.293951287 CYSN 8
HC      H   0.050
SG        0.440719361   10.532225816    1.688457720 CYSN 8
S       S   0.824
LG1      -0.404239097   10.957145937    1.126774557 CYSN 8
LP      L  -0.405
LG2       0.793091788   11.329491558    2.359427872 CYSN 8
LP      L  -0.405
end
```

SEQUENCE LISTING

(1) GENERAL INFORMATION:

(i) APPLICANT: Deem, Michael W.
Rothberg, Jonathan M.
Went, Gregory T.

(ii) TITLE OF INVENTION: CONSENSUS CONFIGURATIONAL BIAS MONTE CARLO METHOD AND SYSTEM FOR PHARMACOPHORE STRUCTURE DETERMINATION

(iii) NUMBER OF SEQUENCES: 10

(iv) CORRESPONDENCE ADDRESS:
(A) ADDRESSEE: Pennie & Edmonds
(B) STREET: 1155 Avenue of the Americas
(C) CITY: New York
(D) STATE: New York
(E) COUNTRY: USA
(F) ZIP: 10036-2711

(v) COMPUTER READABLE FORM:
(A) MEDIUM TYPE: Floppy disk
(B) COMPUTER: IBM PC compatible
(C) OPERATING SYSTEM: PC-DOS/MS-DOS
(D) SOFTWARE: PatentIn Release #1.0, Version #1.30

(vi) CURRENT APPLICATION DATA:
(A) APPLICATION NUMBER: To Be Assigned
(B) FILING DATE: On Even Date Herewith
(C) CLASSIFICATION:

(viii) ATTORNEY/AGENT INFORMATION:
(A) NAME: Misrock, S. Leslie
(B) REGISTRATION NUMBER: 18,872
(C) REFERENCE/DOCKET NUMBER: 7934-007

(ix) TELECOMMUNICATION INFORMATION:
(A) TELEPHONE: (212) 790-9090
(B) TELEFAX: (212) 869-9741/8864
(C) TELEX: 66141 PENNIE

(2) INFORMATION FOR SEQ ID NO:1:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 8 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(ix) FEATURE:
(A) NAME/KEY: Disulfide-bond
(B) LOCATION: 1..8
(D) OTHER INFORMATION: /note= "A disulfide bond is formed between the cysteine residues."

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:1:

```
Cys Xaa Xaa Xaa Xaa Xaa Xaa Cys
1               5
```

```
(2) INFORMATION FOR SEQ ID NO:2:

     (i) SEQUENCE CHARACTERISTICS:
          (A) LENGTH: 102 base pairs
          (B) TYPE: nucleic acid
          (C) STRANDEDNESS: single
          (D) TOPOLOGY: linear

    (ii) MOLECULE TYPE: DNA



    (xi) SEQUENCE DESCRIPTION: SEQ ID NO:2:

ACTTCGAAAT TAATACGACT CACTATAGGG AGACCACAAC GGTTTCCCTC CAGAAATAAT        60

TTTGTTTAAC TTTAACTTTA AGAAGGAGAT ATACATATGC AT                          102

(2) INFORMATION FOR SEQ ID NO:3:

     (i) SEQUENCE CHARACTERISTICS:
          (A) LENGTH: 83 base pairs
          (B) TYPE: nucleic acid
          (C) STRANDEDNESS: single
          (D) TOPOLOGY: linear

    (ii) MOLECULE TYPE: DNA



    (xi) SEQUENCE DESCRIPTION: SEQ ID NO:3:

CCCAGACCCG CCCCCAGCAT TGTGGGTTCC AACGCCCTCT AGACAMNNMN NMNNMNNMNN        60

MNNACAATGT ATATCTCCTT CTT                                                83

(2) INFORMATION FOR SEQ ID NO:4:

     (i) SEQUENCE CHARACTERISTICS:
          (A) LENGTH: 48 base pairs
          (B) TYPE: nucleic acid
          (C) STRANDEDNESS: single
          (D) TOPOLOGY: linear

    (ii) MOLECULE TYPE: DNA



    (xi) SEQUENCE DESCRIPTION: SEQ ID NO:4:

TCGTCTGACC TGCCTCAACC TCCCCACAAT GCTGGCGGCG GCTCTGGT                     48

(2) INFORMATION FOR SEQ ID NO:5:

     (i) SEQUENCE CHARACTERISTICS:
          (A) LENGTH: 42 base pairs
          (B) TYPE: nucleic acid
          (C) STRANDEDNESS: single
          (D) TOPOLOGY: linear

    (ii) MOLECULE TYPE: DNA
```

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:5:

```
ATCAAGTTTG CCTTTACCAG CATTGTGGAG CGCGTTTTCA TC                    42
```

(2) INFORMATION FOR SEQ ID NO:6:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 10 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:6:

```
Met His Cys Xaa Xaa Xaa Xaa Xaa Xaa Cys
1               5                   10
```

(2) INFORMATION FOR SEQ ID NO:7:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 8 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:7:

```
Cys Gly Gly Gly Gly Gly Gly Cys
1               5
```

(2) INFORMATION FOR SEQ ID NO:8:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 30 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:8:

```
NNKNNKNNKN NKNNKNNKNN KNNKNNKNNK                                  30
```

(2) INFORMATION FOR SEQ ID NO:9:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 47 base pairs
(B) TYPE: nucleic acid
(C) STRANDEDNESS: single
(D) TOPOLOGY: linear

(ii) MOLECULE TYPE: DNA

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:9:

```
ACTTCGAAAT TAATACGACT CACTATAGGG AGACCACAAC GGTTTCC                    47
```

(2) INFORMATION FOR SEQ ID NO:10:

(i) SEQUENCE CHARACTERISTICS:
(A) LENGTH: 9 amino acids
(B) TYPE: amino acid
(D) TOPOLOGY: unknown

(ii) MOLECULE TYPE: peptide

(xi) SEQUENCE DESCRIPTION: SEQ ID NO:10:

```
Cys Asn Thr Leu Lys Gly Asp Cys Gly
1               5
```

WHAT IS CLAIMED IS:

1. A method of determining a consensus pharmacophore structure comprising the steps of:
   (a) identifying from one or more diversity libraries a plurality of compounds that bind to a target molecule,
   (b) measuring one or more distances in one or more of the compounds, and
   (c) determining a consensus pharmacophore structure for the compounds.

2. The method of claim 1 wherein said compounds are peptides, peptide derivatives, or peptide analogs.

3. The method of claim 2 wherein said compounds are peptides containing one or more cystines.

4. The method of claim 3 wherein the peptides comprise the sequence CX_6C (SEQ ID NO:1).

5. The method of claim 1 further comprising a step of selecting a plurality of candidate pharmacophores based on rules of chemical homology, the selected plurality of candidate pharmacophores being used in step (c) to determine the consensus pharmacophore structure.

6. The method of claim 5 wherein the rules of homology determine that two candidate pharmacophores are homologous if they have chemically similar side chains.

7. The method of claim 1 which further comprises after said identifying step, a screening step involving a genetic selection technique.

8. The method of claim 1 wher in the step of measuring distance comprises making solid phase nuclear magnetic

resonance measurements on selected nuclei in a nuclear magn tic resonance spectrometer upon a sample comprising on of the compounds.

9. The method of claim 8 wherein the step of measuring distances further comprises making rotational echo double resonance nuclear magnetic resonance measurements of internuclear dipole-dipole interaction strength between selected nuclei in the compound in the sample.

10. The method of claim 8 wherein the sample further comprises a substrate having a surface to which the compound is attached.

11. The method of claim 8 wherein the sample is cooled below room temperature.

12. The method of claim 8 wherein the compound is bound to the target molecule.

13. The method of claim 10 wherein a plurality of the compound is attached to the surface at a surface density such that the inter-nuclear dipole-dipole interactions between different molecules is less than 10% of the inter-nuclear dipole-dipole interaction within one molecule.

14. The method of claim 10 wherein the substrate has pores of sufficient size to permit the target to diffuse and bind to the compound in the sample.

15. The method of claim 9 wherein rotational echo double resonance nuclear magnetic resonance measurements can be made on the compound bound to the target or hydrated or in a dry nitrogen atmosphere.

16. The method of claim 10 wherein the compound is a peptide, and a plurality of the peptide is attached to the substrate surface, which has a purity of the peptide of at least 95% and wherein the surface density of the peptide is no more than one peptide per 100 Å^2 of substrate surface.

17. The method of claim 10 wherein the substrate is selected from the group consisting of p-MethylBenzhydrilamine resin, divinylbenzyl polystyrene resin, and glass beads.

18. The method of claim 8 wherein the selected nuclei are selected from the group consisting of ^{13}C, ^{15}N, ^{19}F, and ^{31}P.

19. The method of claim 9 wherein the nuclear magnetic resonance spectrometer comprises magnetic excitation means, a sample rotor, and free induction decay observing means, and the step of making rotational echo double resonance nuclear magnetic resonance measurements further comprises the steps of:
(a) spinning the sample in the sample rotor,
(b) initially exciting magnetically the selected nuclei to be observed,
(c) providing subsequently one π spin flip magnetic excitation during each rotor period to each of the selected nuclei, the pulses to the different nuclei having fixed phase delays,
(d) observing the free induction decay signal as a function of the number of rotor periods; and
(e) finding the dipole-dipole strength between the selected nuclei, whereby the internuclear distance between the selected nuclei can be obtained.

20. The method of claim 1 wherein the step of measuring distances comprises making liquid phase nuclear magnetic resonance measurements.

21. A method of determining a consensus pharmacophore structure comprising the st ps of:
   (a) identifying from one or more diversity libraries a plurality of compounds that bind to a target molecule,
   (b) determining a consensus pharmacophore structure for the compounds.

22. A method of determining a consensus pharmacophore structure comprising the steps of:
   (a) measuring one or more distances in one or more compounds that bind to a target molecule, and
   (b) determining a consensus pharmacophore structure for the compounds.

23. The method of claim 21 or 22 further comprising a step of selecting a plurality of candidate pharmacophores based on rules of chemical homology, the selected plurality of candidate pharmacophores being used in step (b) to determine the consensus pharmacophore structure.

24. The method of claim 23 wherein the compounds have limited conformational degrees of freedom at the temperature of interest, and wherein the step of determining a consensus pharmacophore structure for each compound further comprises, performing a consensus configurational bias Monte Carlo method, said Monte Carlo method comprising the steps of:
   (a) generating a proposed structure for a compound identified from said one or more diversity libraries by making conformational alterations consistent with the conformational degrees of freedom, the alterations being made to a representation of the compound's current chemical and conformational structure to generate a proposed representation, the proposed structure being g n rated with a bias

toward more acceptable configurations of lower en rgy, wh reby the method is made more efficient,

(b) acc pting and st ring the proposed structure according to a probability depending on an energy determined for the proposed structure, and

(c) repeating these steps until sufficient structures have been stored for each compound to permit statistically significant determination of an equilibrium structure for each compound.

25. A method of determining one or more lead compounds for use as a drug that binds to a target molecule comprising the steps of:

(a) identifying from one or more diversity libraries a plurality of compounds that bind to a target molecule;

(b) determining a consensus pharmacophore structure for the compounds; and

(c) determining one or more lead compounds for use as a drug which share a pharmacophore specification with the determined consensus pharmacophore structure.

26. A method of determining one or more lead compounds for use as a drug that binds to a target molecule comprising the steps of:

(a) measuring one or more distances in one or more compounds that bind to a target molecule;

(b) determining a consensus pharmacophore structure for the compounds; and

(c) determining one or more lead compounds for use as a drug which share a pharmacophore specification with the determined consensus pharmacophore structure.

27. The method according to claim 25 or 26 wherein said step of determining one or more lead compounds comprises modifying a compound id ntified as binding to th targ t mol cul , said modification being done outside of the

pharmacophore structure, to rend r the compound more attractive for use as a drug.

28. The method of claim 5 wherein the compounds have limited conformational degrees of freedom at a temperature of interest, and wherein the step of determining a consensus pharmacophore structure for the compounds further comprises performing a consensus configurational bias Monte Carlo method, said Monte Carlo method comprising the steps of:
(a) generating a proposed structure for a compound identified from said one or more diversity libraries by making conformational alterations consistent with the conformational degrees of freedom, the alterations being made to a representation of the compound's current chemical and conformational structure to generate a proposed representation, the proposed structure being generated with a bias toward more acceptable configurations of lower energy,
(b) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and
(c) repeating these steps until sufficient structures have been stored for each compound to permit statistically significant determination of an equilibrium structure for each compound.

29. The method of claim 28 wherein the limited conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

30. The method of claim 28 wherein the compound is a peptide, p ptide derivative, or peptide analog.

31. The method of claim 28 wherein the conformational alterations comprise constrained, concerted torsional rotations or removal of a side chain and regrowth of the side chain with a new torsional conformation.

32. The method of claim 31 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

33. The method of claim 28 wherein determining the energy for the proposed structure of one compound comprises including one or more constraint terms which represent knowledge of measured structure for the compound.

34. The method of claim 33 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

35. The method of claim 28 wherein the energy is determined for the proposed structure of one compound by a method comprising including consensus terms which represent knowledge that the identified compounds all bind to the same target, the compounds being otherwise treated independently by the method.

36. The method of claim 35 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of a candidate pharmacophore from the average values of these positions in all the compounds.

37. The method of claim 35 wherein the step of determining the consensus pharmacophore structure comprises determining from the plurality of selected candidate pharmac phores a candidate pharmacophore for which the

consensus terms are relatively small compared to the total energy.

38. The method of claim 35 wherein the step of determining the consensus pharmacophore structure comprises determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are minimum compared to other selected regions.

39. The method of claim 28 wherein the equilibrium structure is determined by a method comprising averaging selected generated and accepted structures for each compound.

40. The method of claim 39 wherein the averaging of structures comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets for each member.

41. A method of identifying a compound that binds to a target molecule comprising the following steps in the order stated:
(a) contacting compounds of a phage display or polysome-based diversity library with a target molecule;
(b) identifying one or more compounds in the library that bind to the target molecule;
(c) contacting one or more first fusion proteins, each first fusion protein comprising an identified compound, with a second fusion protein comprising the target molecule or a binding portion thereof, in which binding of the first fusion protein to the second fusion protein results in an increase in activity or activation of a transcriptional promoter or an origin of replication; and
(d) identifying one or more of the compounds that when present in said first fusion protein result in said increase in activity r activation.

42. A m thod of making solid state nuclear magnetic resonanc measurements comprising measuring int rnuclear dipole-dipol interaction strengths between selected nuclei in a compound, said compound being attached to the surface of a substrate.

43. The method of claim 42 which further comprises before said measuring step the step of synthesizing a plurality of said compound on the surface of the substrate.

44. The method of claim 43 wherein said plurality of the compound is at least 95% pure.

45. The method of claim 42 wherein a plurality of said compound is attached to the substrate surface, with at least 10 Å spacing between molecules of the compound.

46. The method of claim 42 wherein the substrate has pores of sufficient size to permit a molecule to diffuse and bind to the compound.

47. The method of claim 42 wherein the substrate has a surface density of the compound such that the inter-nuclear dipole-dipole interactions between different molecules of the compound is less than 10% of the inter-nuclear dipole-dipole interaction within one molecule of the compound.

48. The method of claim 42 wherein the compound is a peptide, peptide derivative, or peptide analog.

49. The method of claim 42 wherein the substrate is selected from the group consisting of p-MethylBenzhydrilamine resin, divinylbenzyl polystyrene resin, and a glass bead.

50. The method of claim 42 wherein said measuring step compris s using a nuclear magnetic resonance

spectrometer, said spectrometer comprising magnetic excitation means, a sample rot r, and free induction decay observing means; and said measurement of internuclear dipole-dipole interaction is done by a method comprising the steps of:

(a) spinning the sample in the sample rotor;

(b) initially exciting magnetically the selected nuclei to be observed;

(c) providing subsequently one or more π spin flip magnetic excitations during each rotor period to one or both of the selected nuclei, wherein pulses to the different nuclei have fixed phase delays;

(d) observing a free induction decay signal as a function of the number of rotor periods; and

(e) determining the dipole-dipole strength between the selected nuclei, whereby the internuclear distance between the selected nuclei can be obtained.

51. A method of configurational bias Monte Carlo determination of the structure of a compound having limited conformational degrees of freedom at a temperature of interest, the method comprising the steps of:

(a) generating a proposed structure for the compound by making conformational alterations consistent with the conformational degrees of freedom, the alterations being made to a representation of the compound's current chemical and conformational structure to generate a proposed representation;

(b) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure; and

(c) repeating these steps until sufficient structures have been stored to permit statistically significant determination of an equilibrium structure.

52. The method of claim 51 wherein the conformational degre s of freedom compris torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

53. The method of claim 51 wherein the compound is a peptide, peptide derivative, or peptide analog.

54. The method of claim 51 wherein the conformational alterations comprise constrained, concerted torsional rotations.

55. The method of claim 54 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

56. The method of claim 51 wherein the conformational alterations comprise removal of a side chain and regrowth of the side chain with a new torsional conformation.

57. The method of claim 51 wherein the proposed structures are generated with a bias toward more acceptable configurations of lower energy.

58. The method of claim 51 wherein the energy is determined for the proposed structure by a method comprising including constraint terms which represent knowledge of measured structure for the compound.

59. The method of claim 58 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

60. The method of claim 51 applied to a plurality of compounds f limited c nf rmational degrees of freedom all of which bind to th sam target molecule wherein the method further comprises a step of selecting a plurality of candidate pharmacophores based on rules of chemical homology.

61. The method of claim 60 wherein the energy is determined for the proposed structure of one of the plurality of compounds by a method comprising including consensus terms which represent knowledge that the compounds all bind to the same target molecule.

62. The method of claim 61 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of a candidate pharmacophore of said one of the plurality of compounds from the average values of these positions in all the compounds.

63. The method of claim 61 which further comprises a step of determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores that candidate pharmacophore for which the consensus terms are minimum compared to other candidate pharmacophores.

64. The method of claim 60 which further comprises a step of determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores that candidate pharmacophore for which the consensus terms are relatively small compared to the total energy.

65. The method of claim 63 or 64 which further comprises a step of determining one or more lead compounds for use as a drug which shar a pharmacophore specification with th d termined consensus pharmacophor structure.

66. The method of claim 51 wherein the equilibrium structure is det rmined by a method comprising averaging selected generated and accepted structures.

67. The method of claim 66 wherein the averaging of structures comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets.

68. An apparatus for configurational bias Monte Carlo determination of the structure of a compound having limited conformational degrees of freedom at a temperature of interest, the apparatus comprising:

(a) memory means for storing

 (i) data structures representing the compound's chemical and conformational structure consistently with the compound's degrees of freedom,

 (ii) similar data structures representing the compound's proposed structure and prior structures, and

 (iii) parameters representing atomic interactions, and

(b) processor means for executing programs for

 (i) generating a proposed structure by making conformational alterations consistent with the conformational degrees of freedom and with a bias toward more acceptable configurations of lower energy,

 (ii) accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and

 (iii) repeating these steps until sufficient structures have been stored to permit statistically significant determination of an equilibrium structure.

69. The apparatus of claim 68 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

70. The apparatus of claim 68 wherein the compound is a peptide, peptide derivative, or peptide analog.

71. The apparatus of claim 68 wherein the memory, processor, and control means are configured from a workstation typ digital computer comprising RAM memory, disk memory, processor, and input and display devices.

72. The apparatus of claim 68 wherein the conformational alterations made by the processor means further comprise constrained, concerted torsional rotations or removal of a side chain and regrowth of the side chain with a new torsional conformation.

73. The apparatus of claim 72 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

74. The apparatus of claim 68 wherein the processor means determines an energy for the proposed structure by a method comprising including constraint terms which represent knowledge of measured structure for the compound.

75. The apparatus of claim 74 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

76. The apparatus of claim 68 applied to a plurality of comp unds of limited conformational degrees of fre dom all of which bind to the sam target molecule, and wherein the processor means further comprises programs for selecting a plurality of candidate pharmacophores based on rules of chemical homology.

77. The apparatus of claim 76 wherein the processor means determines an energy for the proposed structure of any one compound by a method comprising including consensus terms which represent knowledge that the compounds all bind to the same target molecule.

78. The apparatus of claim 77 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of the candidate pharmacophore of said one compound from the average values of these positions in all the compounds.

79. The apparatus of claim 77 wherein the processor means further comprises programs for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are minimum compared to other candidate pharmacophores.

80. The apparatus of claim 77 wherein the processor means further comprises programs for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are relatively small compared to the total energy.

81. The apparatus of claim 79 or 80 wherein the processor means further comprises programs for determining ne or more lead compounds for use as a drug that share a

pharmacophore specification with the cons nsus pharmac ph re structure.

82. The apparatus of claim 68 wherein the processor means determines an equilibrium structure by a method comprising averaging selected generated and accepted structures.

83. The apparatus of claim 82 wherein the averaging of structures further comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets.

84. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of at least one compound having limited conformational degrees of freedom at a temperature of interest, said apparatus comprising:

(a) first memory means for storing data structures representing the compound's chemical and conformational structure consistently with the compound's degrees of freedom,

(b) second memory means for storing similar data structures representing the compound's proposed structure,

(c) third memory means for storing similar data structures representing the compound's prior structures,

(d) first processor means for generating a proposed structure by making conformational alterations consistent with the conformational degrees of freedom and with a bias toward conformations of lower energy,

(e) second processor means for accepting and storing the proposed structure according to a probability depending on an energy determined for the proposed structure, and

(f) third processor means for controlling and repeating th generation and acceptance until suffici nt structures have be n stored to permit statistically significant determination of an equilibrium structure.

85. The digital computer apparatus of claim 84 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

86. The digital computer apparatus of claim 84 wherein the compound is a peptide, peptide derivative, or peptide analog.

87. The digital computer apparatus of claim 84 wherein the digital computer is a workstation type digital computer comprising RAM memory, disk memory, processor, and input and display devices.

88. The digital computer apparatus of claim 84 wherein the conformational alterations generated by the first processor means comprise constrained, concerted torsional rotations or removal of a side chain and regrowth of the side chain with a new torsional conformation.

89. The digital computer apparatus of claim 88 wherein the constrained, concerted torsional rotations are constrained so that no more than four rigid units are spatially displaced.

90. The digital computer apparatus of claim 84 wherein the second processor means determines an energy for the

proposed structure by a method comprising including constraint terms which represent knowledge of measured structure for the compound.

91. The digital computer apparatus of claim 90 wherein the constraint terms comprise a weighted sum of squares of differences of the actual and measured structures.

92. The digital computer apparatus of claim 84 in which said at least one compound is a plurality of compounds of limited conformational degrees of freedom all of which bind to the same target and wherein data are stored in said first memory means representing the chemical and conformational structure of said plurality of compounds and wherein the apparatus further comprises additional processor means for selecting a plurality of candidate pharmacophores based on rules of chemical homology.

93. The digital computer apparatus of claim 92 wherein the second processor means determines an energy for the proposed structure of one of said plurality of compounds by a method comprising including consensus terms which represent knowledge that the compounds all bind to the same target molecule.

94. The digital computer apparatus of claim 92 wherein the consensus terms are a weighted sum of squares of differences in the atomic positions of a candidate pharmacophore of said one of the plurality of compounds from the average values of these positions in all the compounds.

95. The digital computer apparatus of claim 93 wherein the apparatus further comprises processor means for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacoph r s a candidat pharmacophore for which the

consensus terms are relatively small compared to th total energy.

96. The digital computer apparatus of claim 93 wherein the apparatus further comprises processor means for determining a consensus pharmacophore structure by determining from the plurality of selected candidate pharmacophores a candidate pharmacophore for which the consensus terms are minimum compared to other candidate pharmacophores.

97. The digital computer apparatus of claims 95 or 96 wherein the apparatus further comprises processor means for determining one or more lead compounds for use as a drug that share a pharmacophore specification with the consensus pharmacophore structure.

98. The digital computer apparatus of claim 84 wherein the third processor means determines an equilibrium structure by a method comprising averaging selected generated and accepted structures.

99. The digital computer apparatus of claim 98 wherein the averaging of structures comprises clustering selected generated and accepted structures into sets of similar structures and averaging these sets.

100. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of a plurality of compounds having limited conformational degrees of freedom, each compound having a backbone and side chains, said apparatus comprising:

(a) first memory means for storing data structures representing each compound's chemical and conformational structure consistently with that compound's degrees of freedom and constraints,

(b) second mem ry m ans f r storing similar data structures representing a proposed structure for one or more of the compounds,

(c) third memory means for storing similar data structures representing prior structures of the plurality of compounds,

(d) first processor means for generating a proposed structure of a randomly selected compound by making conformational alterations consistent with the conformational degrees of freedom, the conformational alterations being randomly distributed between alterations that alter the structure of a randomly selected side chain of the selected compound and alterations that alter the structure of a randomly selected region of the backbone of the selected compound, the proposed structure being stored in the second memory means, the proposed structure being generated with a bias toward more acceptable structures of lower energy, whereby the method is made more efficient,

(e) second processor means for accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, the energy including terms representing physical interactions and terms representing heuristic information about the compound's structure, the heuristic information comprising knowledge about measured distances in one or more compounds of said plurality and about the plurality of the compounds binding to a same target molecule,

(f) third processor means for controlling and repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

101. The digital computer of claim 100 wherein the conformational degrees of freedom comprise torsional rotations about mutual bonds between otherwise rigid subunits of the compound, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position, the torsional rotations respecting any conformational constraints present.

102. The digital computer of claim 100 wherein the compound is a peptide, peptide derivative, or peptide analog.

103. A method of configurational bias Monte Carlo determination of the structure of a compound selected from the group consisting of a peptide, peptide derivative, and peptide analog, the method comprising the steps of:

(a) representing the conformation of the compound by interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(b) generating a proposed structure by making conformational alterations consistent with the compound's structure,

(c) accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, and

(d) repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

104. An apparatus for configurational bias Monte Carlo determination of the structure f a compound selected

fr m the group consisting of a peptide, peptide derivative, and peptide analog, the apparatus c mprising:

(a) memory means for storing
    (i) data structures representing the compound's conformation as interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,
    (ii) similar data structures representing the compound's proposed structure and prior structures, and
    (iii) parameters representing atomic interactions, and
(b) processor means for executing programs for
    (i) generating a proposed structure by making conformational alterations consistent with the compound's structure and with a bias toward more acceptable configurations of lower energy,
    (ii) accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, and
    (iii) repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

105. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of a compound selected from the group consisting of a peptide, peptide derivative, and peptide analog, said apparatus comprising:

(a) first memory means for storing data structures representing the compound's structure as interconnected rigid units capable of t rsional rotation about common b nds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(b) second memory means for storing similar data structures representing the compound's proposed structure,

(c) third memory means for storing similar data structures representing the compound's prior structures,

(d) first processor means for generating a proposed structure by making conformational alterations consistent with the compound's structure and constraints and with a bias toward conformations of lower energy,

(e) second processor means for accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, and

(f) third processor means for controlling and repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

106. In a digital computer, apparatus for configurational bias Monte Carlo determination of the structure of a plurality of compounds selected from the group consisting of peptides, peptide derivatives, and peptide analogs, each compound having a backbone and side chains, said apparatus comprising:

(a) first memory means for storing data structures representing each compound's structure as interconnected rigid units capable of torsional rotation about common bonds, each rigid unit's representation comprising its interconnections and atomic composition, each atom's representation comprising its type and position,

(b) s cond memory m ans for storing similar data structures representing a proposed structure for one or more of th compounds,

(c) third memory means for storing similar data structures representing prior structures of the plurality of the compounds,

(d) first processor means for generating a proposed structure of a randomly selected compound by making conformational alterations consistent with the compound's structure, the conformational alterations being randomly distributed between alterations that alter the structure of a randomly selected side chain of the selected compound and alterations that alter the structure of a randomly selected region of the backbone of the selected compound, the proposed structure being stored in the second memory means, the proposed structure being generated with a bias toward more acceptable structures of lower energy,

(e) second processor means for accepting a proposed structure according to a probability depending on an energy determined for the proposed structure, the energy including terms representing physical interactions and terms representing heuristic information about the compound's structure, the heuristic information comprising knowledge about measured distances in one or more compounds of said plurality and about the plurality of the compounds binding to a same target molecule,

(f) third processor means for controlling and repeating these steps until sufficient structures have been generated and accepted to permit statistically significant determination of an equilibrium structure.

107. The method of claim 42 wherein the nuclear magnetic res nance is rotational echo double r sonance.

108. The method of claim 1 wherein the diversity libraries are structurally constrained organic diversity libraries.

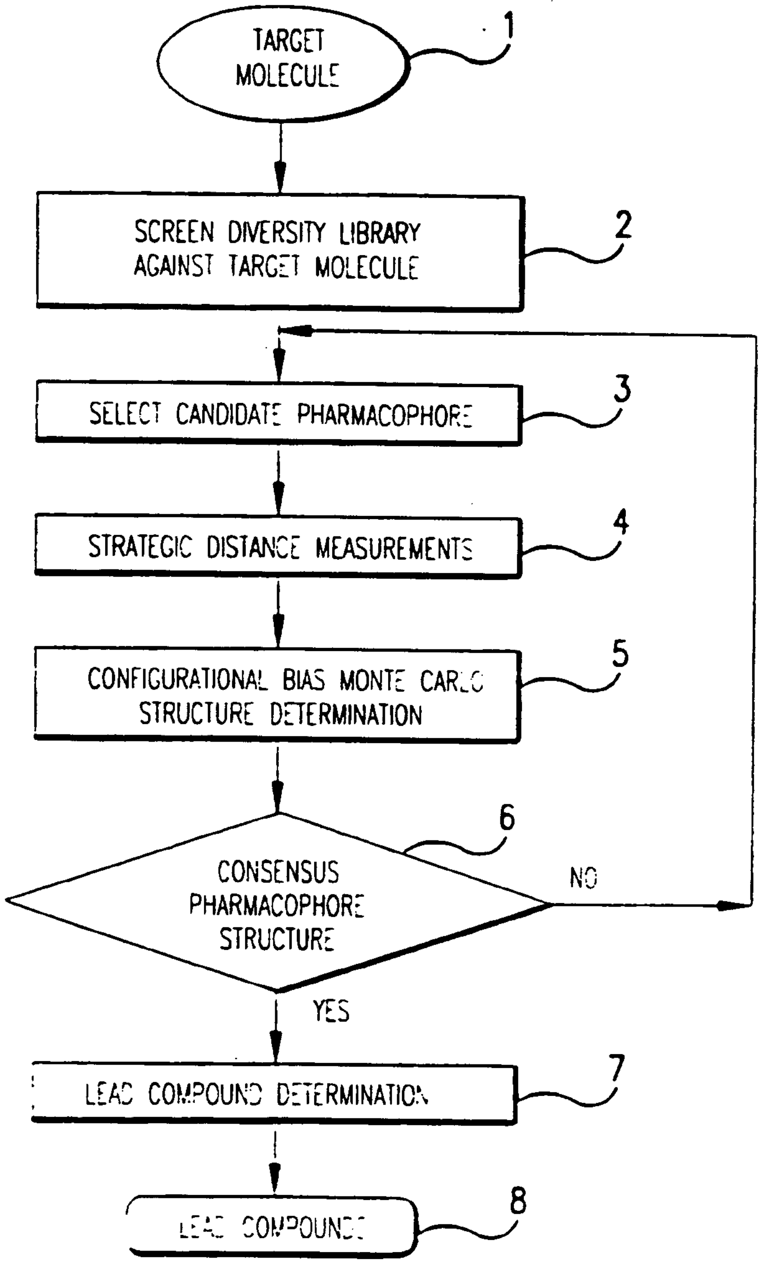

TARGET MOLECULE
1
SCREEN DIVERSITY LIBRARY AGAINST TARGET MOLECULE
2
SELECT CANDIDATE PHARMACOPHORE
3
STRATEGIC DISTANCE MEASUREMENTS
4
CONFIGURATIONAL BIAS MONTE CARLO STRUCTURE DETERMINATION
5
CONSENSUS PHARMACOPHORE STRUCTURE
6
NO
YES
LEAD COMPOUND DETERMINATION
7
LEAD COMPOUND
8

FIG.1

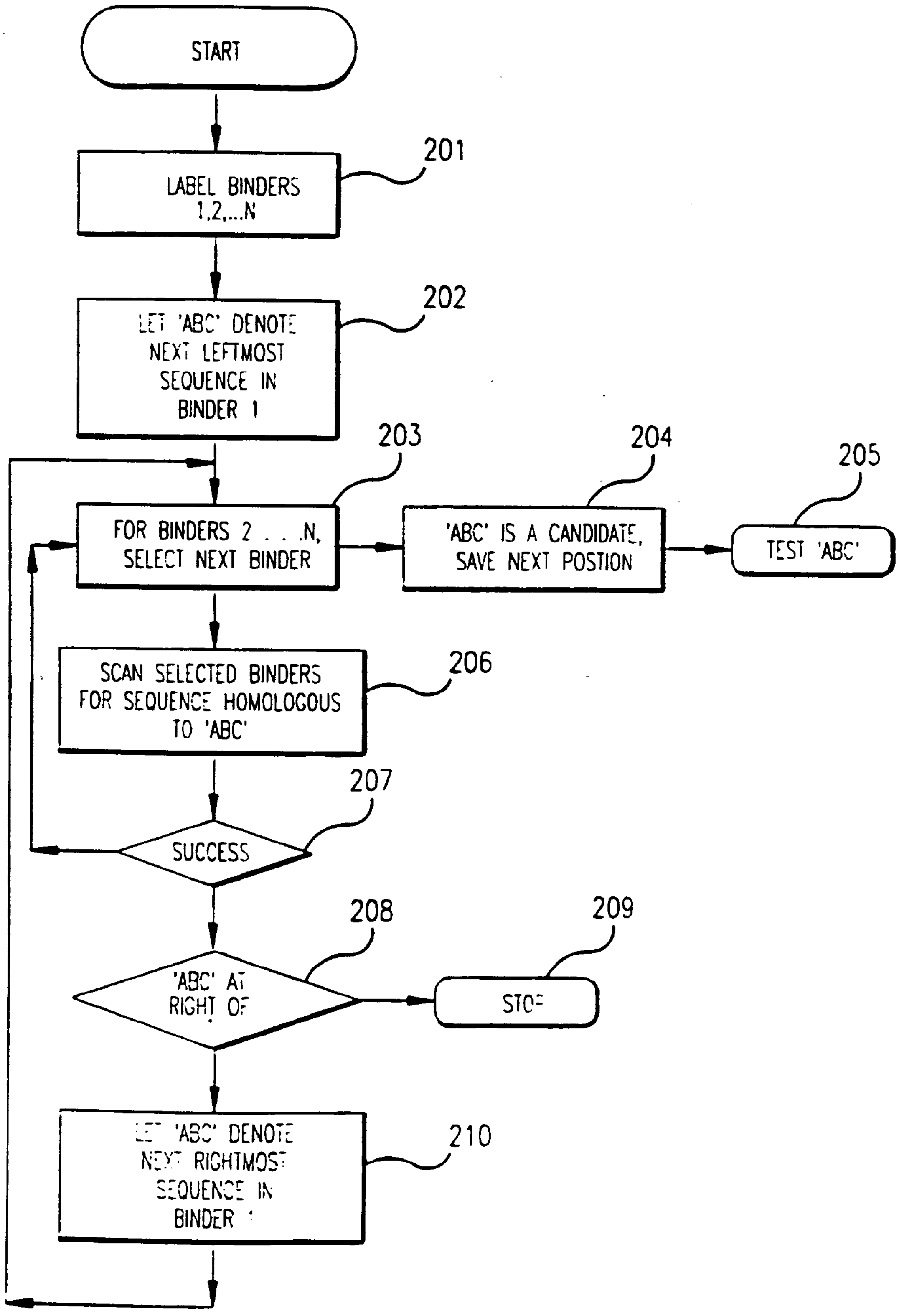

START
201
LABEL BINDERS 1,2,...N
202
LET 'ABC' DENOTE NEXT LEFTMOST SEQUENCE IN BINDER 1
203
FOR BINDERS 2 . . .N, SELECT NEXT BINDER
204
'ABC' IS A CANDIDATE, SAVE NEXT POSTION
205
TEST 'ABC'
206
SCAN SELECTED BINDERS FOR SEQUENCE HOMOLOGOUS TO 'ABC'
207
SUCCESS
208
'ABC' AT RIGHT OF
209
STOP
210
LET 'ABC' DENOTE NEXT RIGHTMOST SEQUENCE IN BINDER 1

FIG.2A

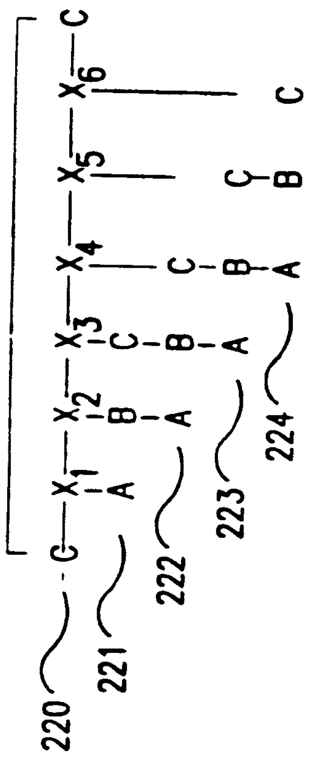


FIG.2B

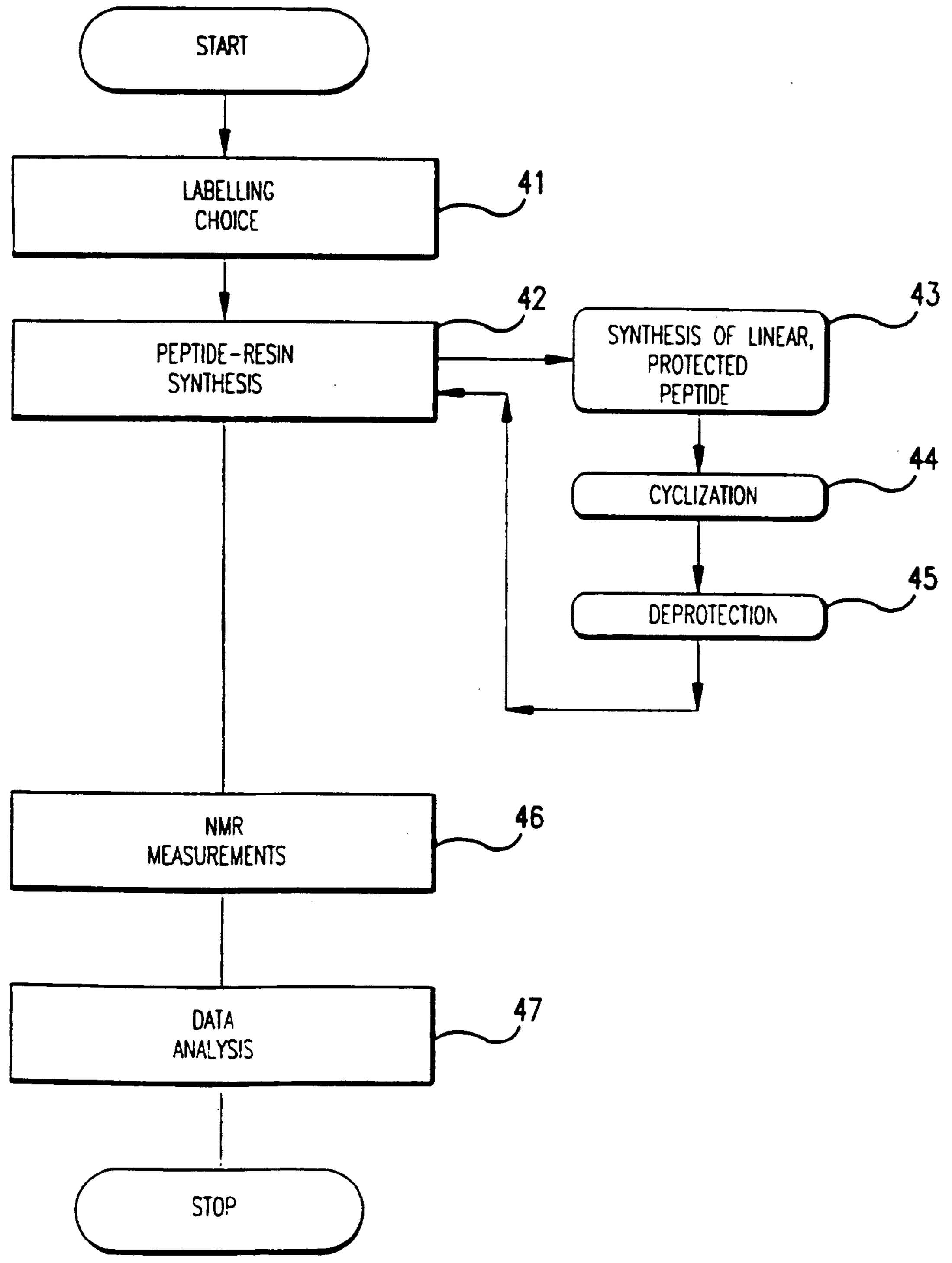

START
LABELLING CHOICE
41
PEPTIDE-RESIN SYNTHESIS
42
SYNTHESIS OF LINEAR, PROTECTED PEPTIDE
43
CYCLIZATION
44
DEPROTECTION
45
NMR MEASUREMENTS
46
DATA ANALYSIS
47
STOP

FIG.3

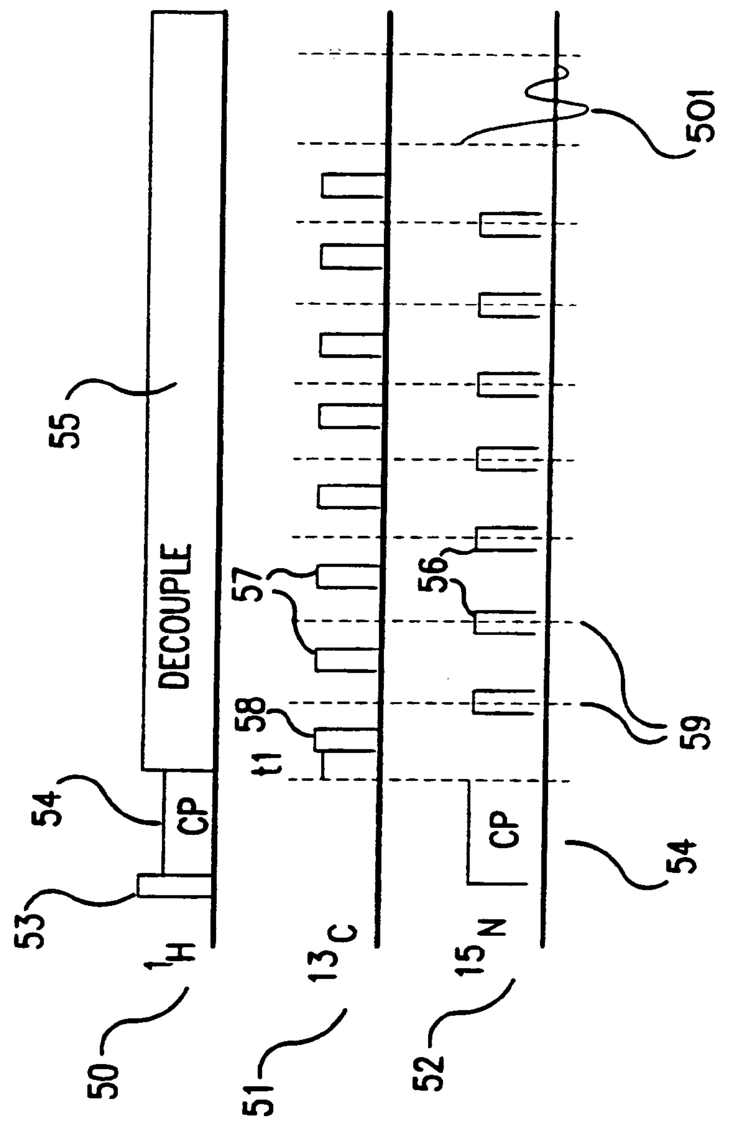


FIG.4

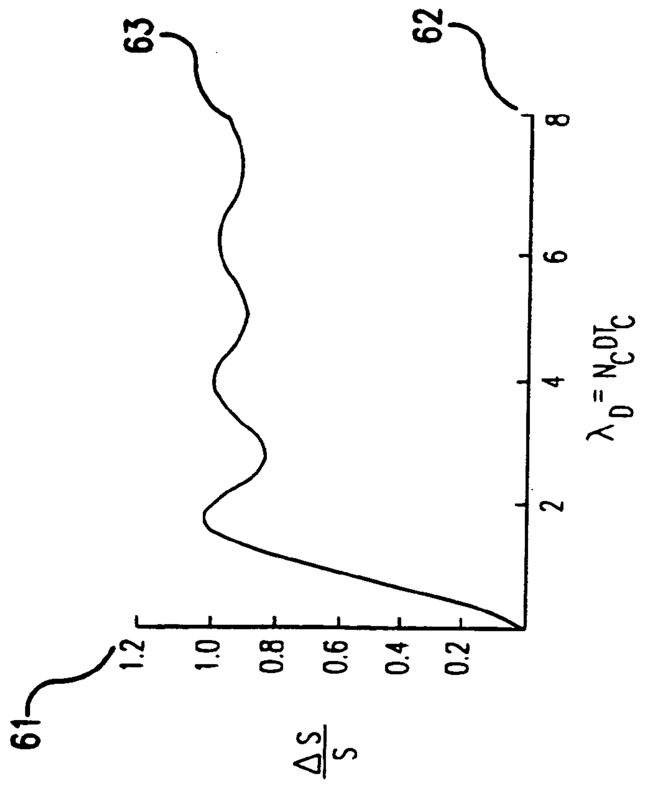


FIG.5

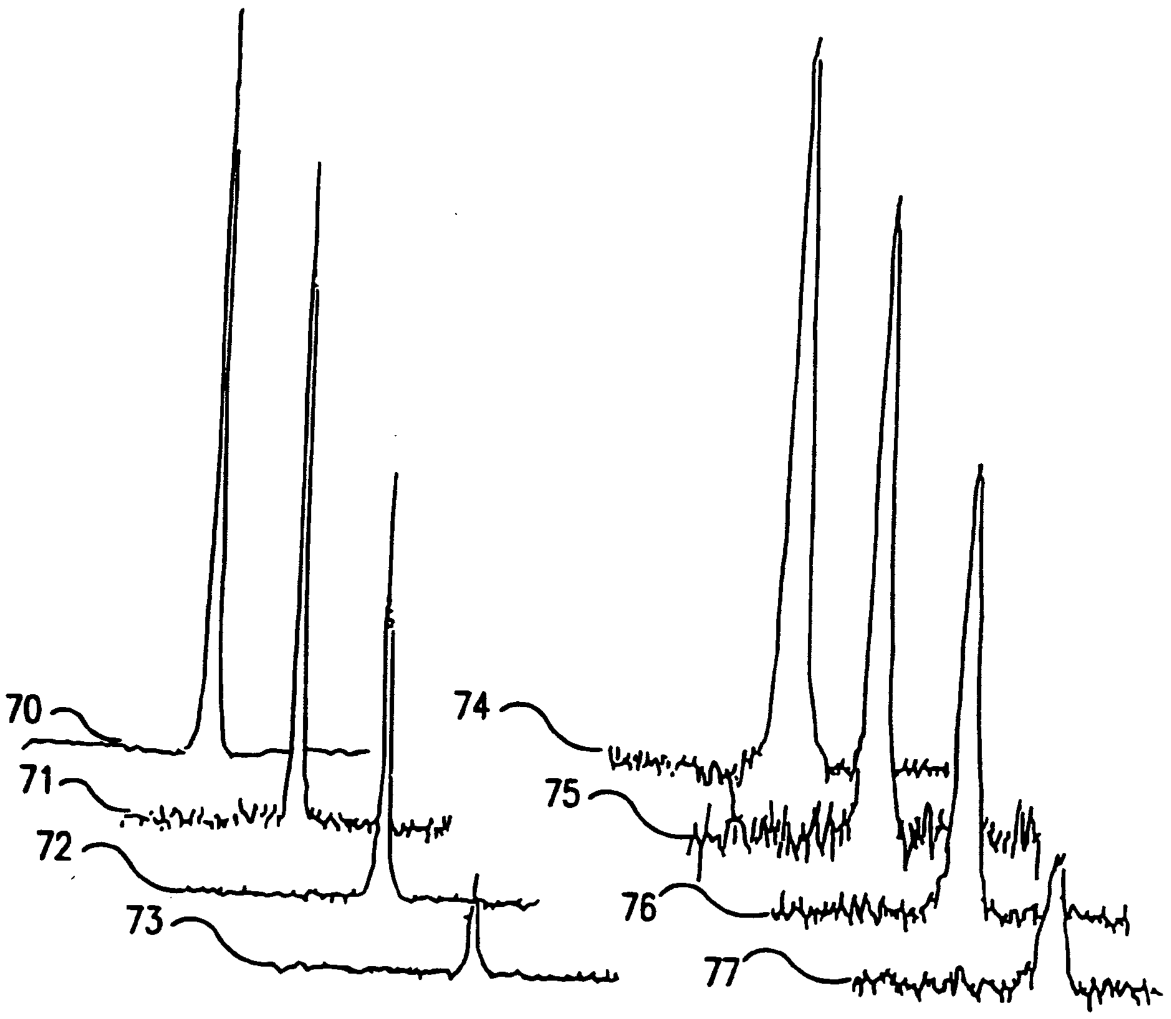


FIG.6

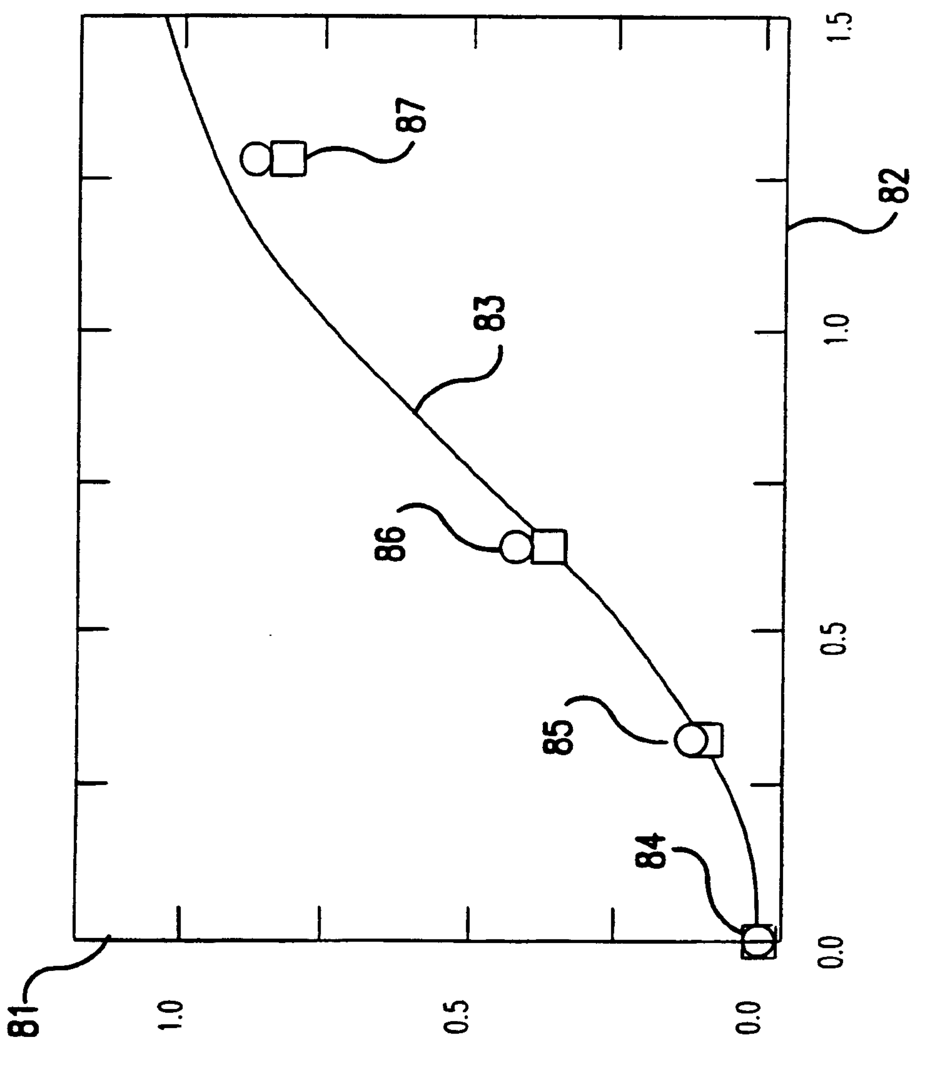


FIG.7

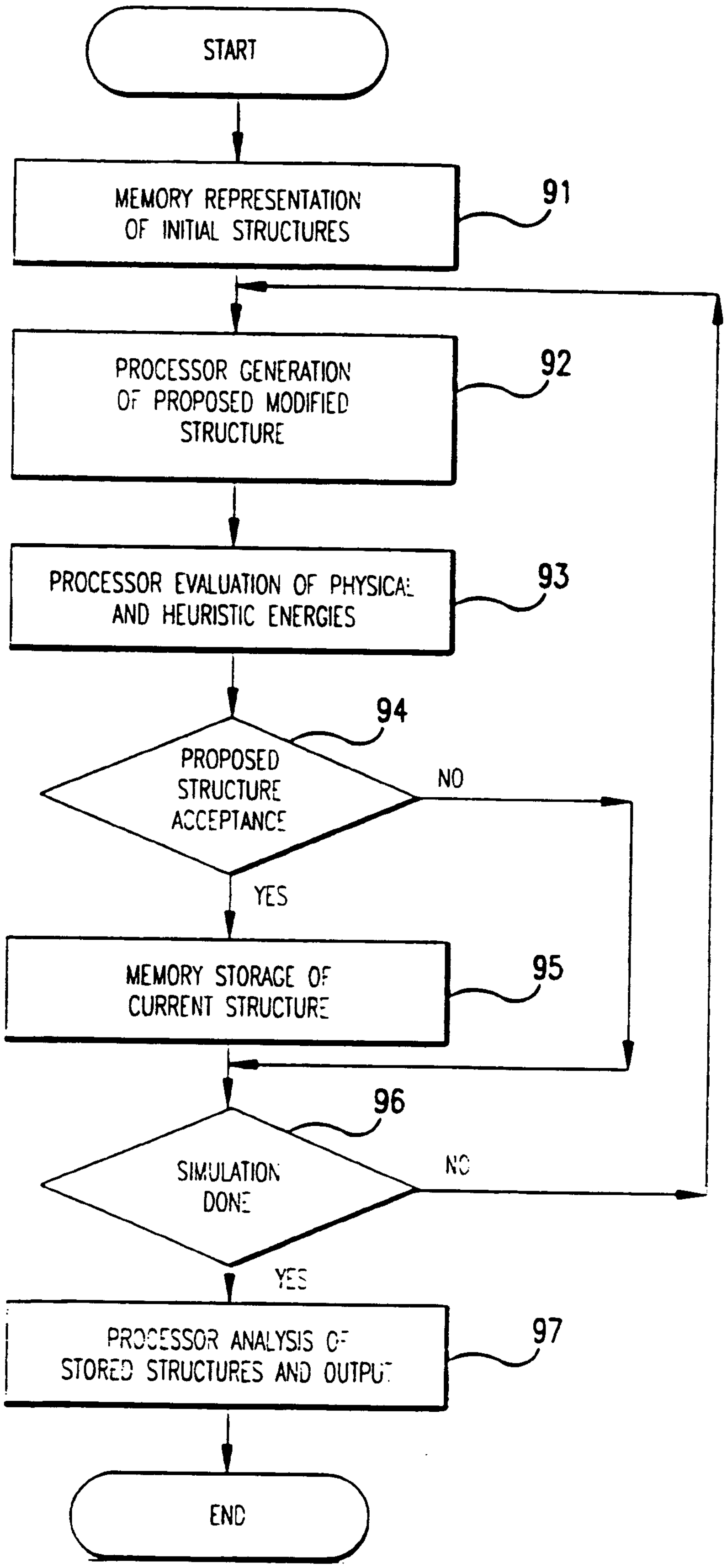

START
MEMORY REPRESENTATION OF INITIAL STRUCTURES
91
PROCESSOR GENERATION OF PROPOSED MODIFIED STRUCTURE
92
PROCESSOR EVALUATION OF PHYSICAL AND HEURISTIC ENERGIES
93
PROPOSED STRUCTURE ACCEPTANCE
94
NO
YES
MEMORY STORAGE OF CURRENT STRUCTURE
95
SIMULATION DONE
96
NO
YES
PROCESSOR ANALYSIS OF STORED STRUCTURES AND OUTPUT
97
END

FIG.8

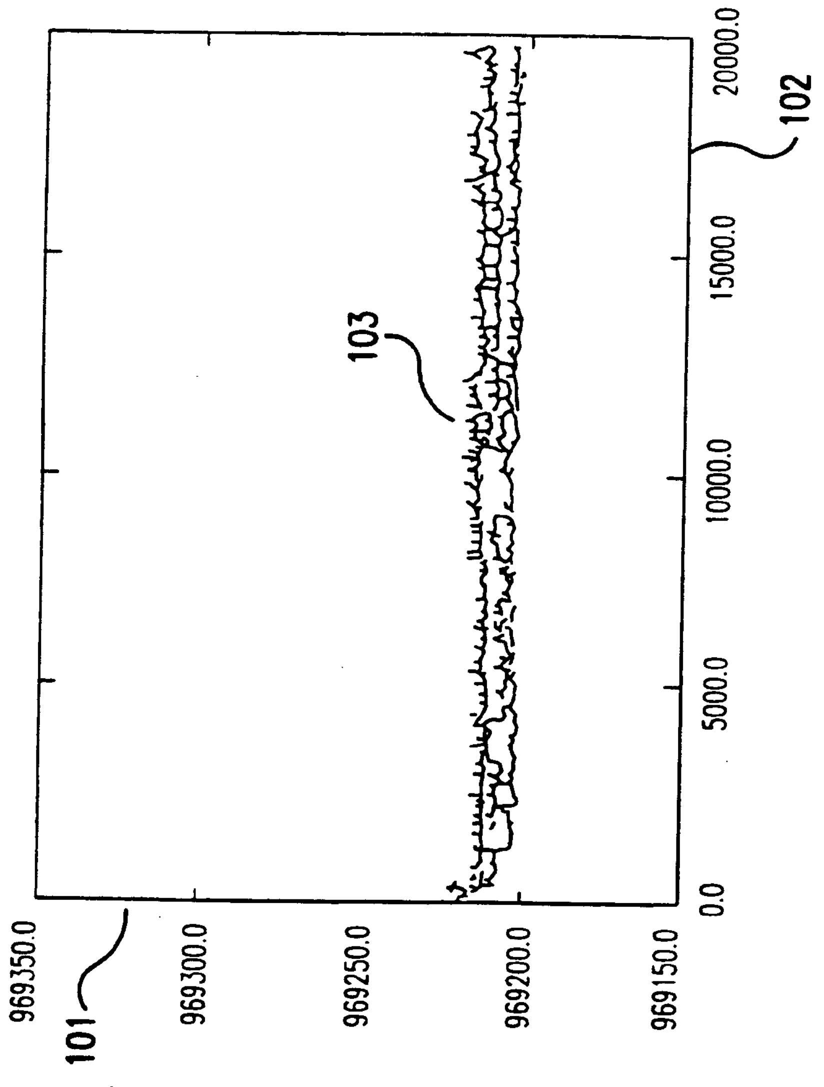

969350.0
969300.0
969250.0
969200.0
969150.0
0.0
5000.0
10000.0
15000.0
20000.0
101
102
103

FIG.9

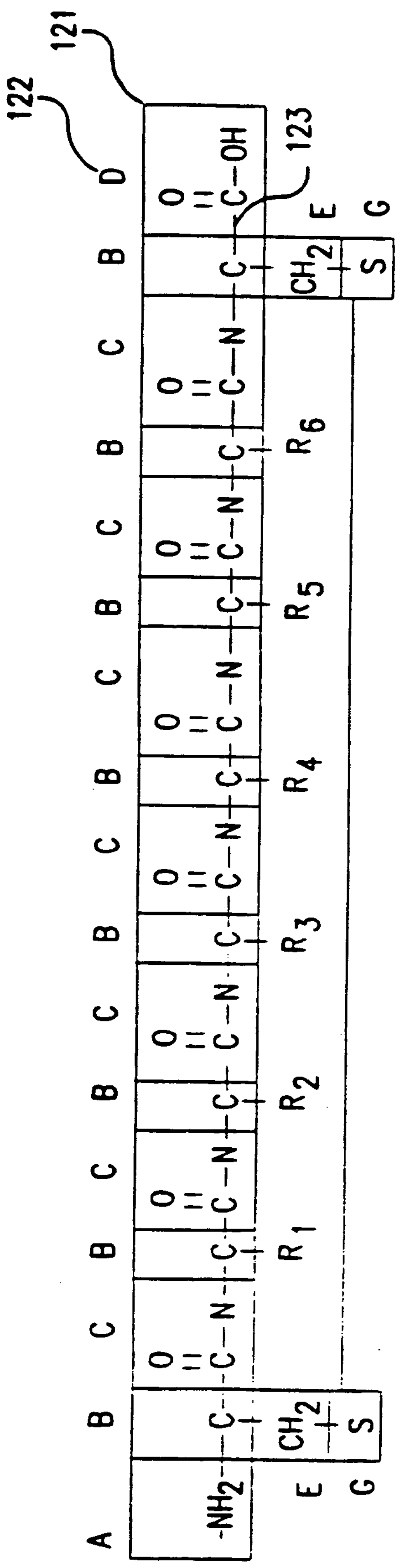
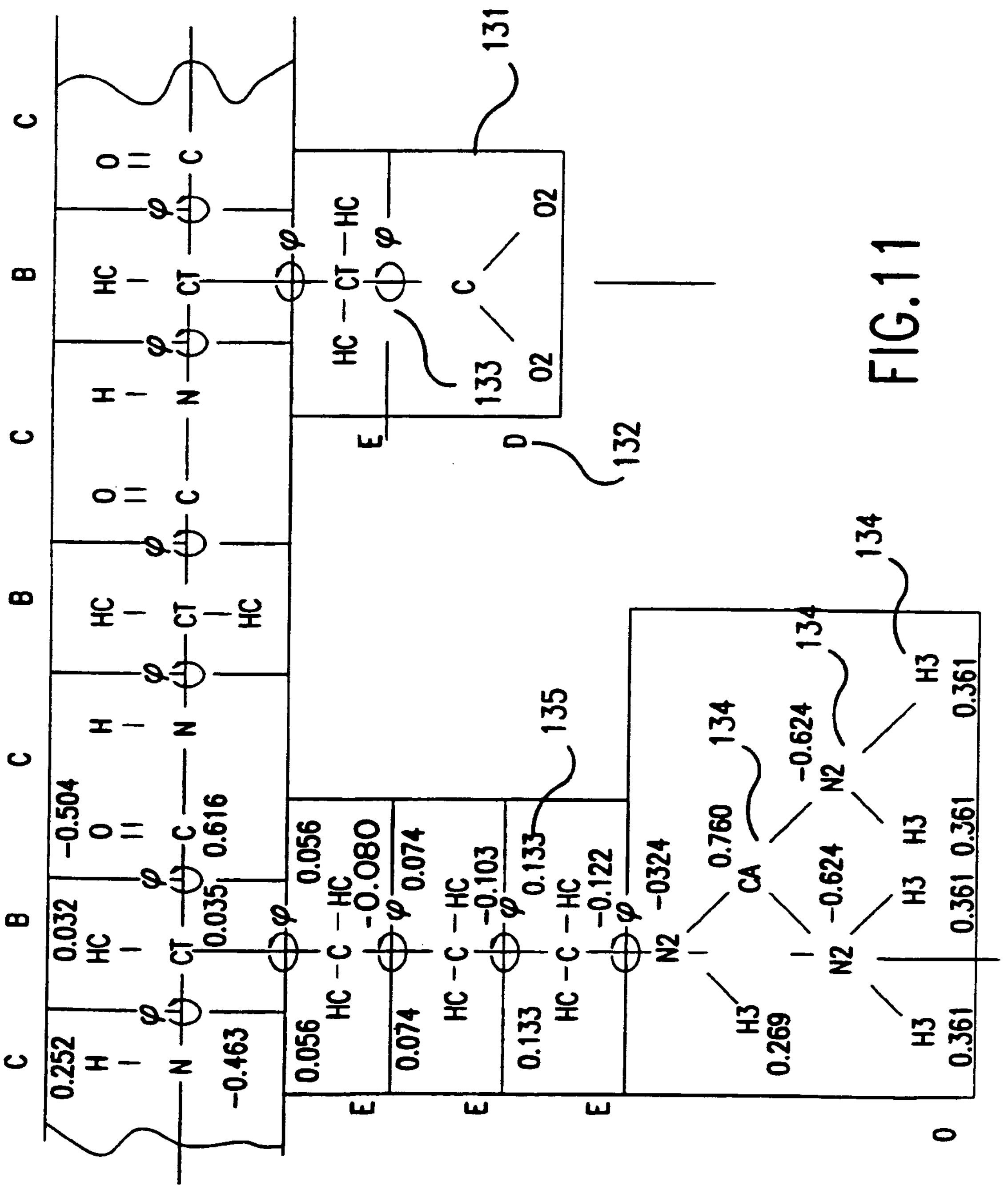


FIG.10



C
B
C
B
C
B
C
D
E
131
132
133
134
135
0.252
-0.463
0.032
0.035
-0.504
0.616
0.056
-0.080
0.074
-0.103
0.133
-0.122
-0324
0.760
-0.624
0.269
0.361

FIG.11

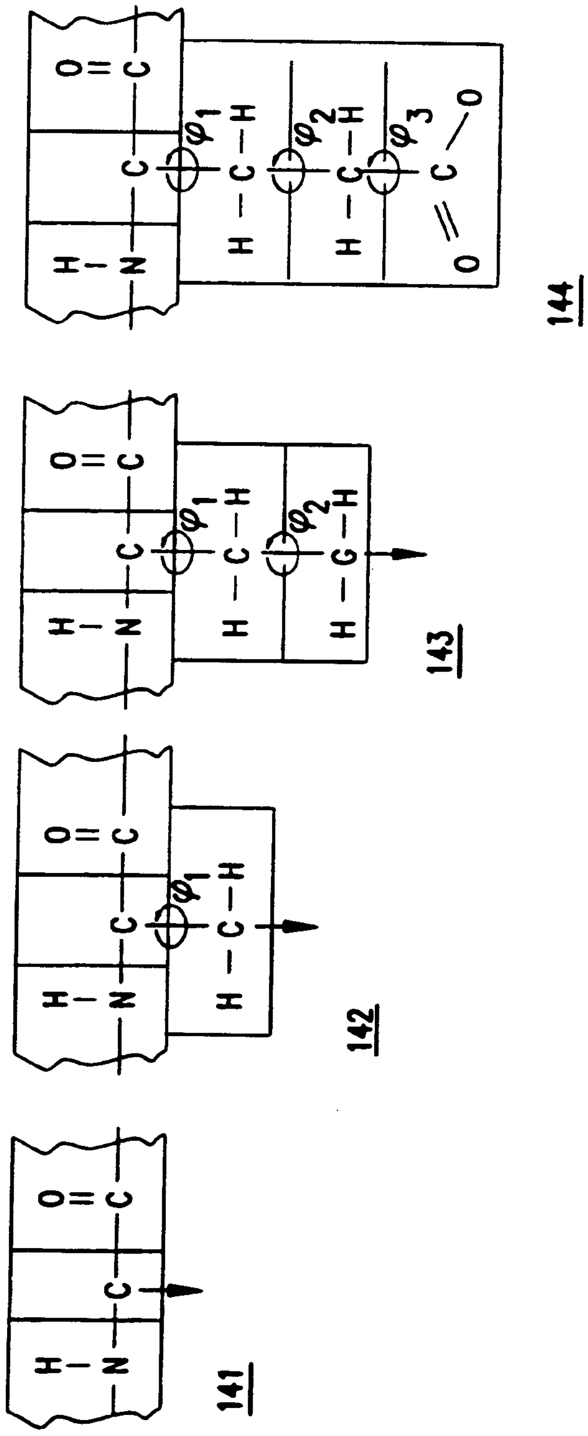


FIG.12

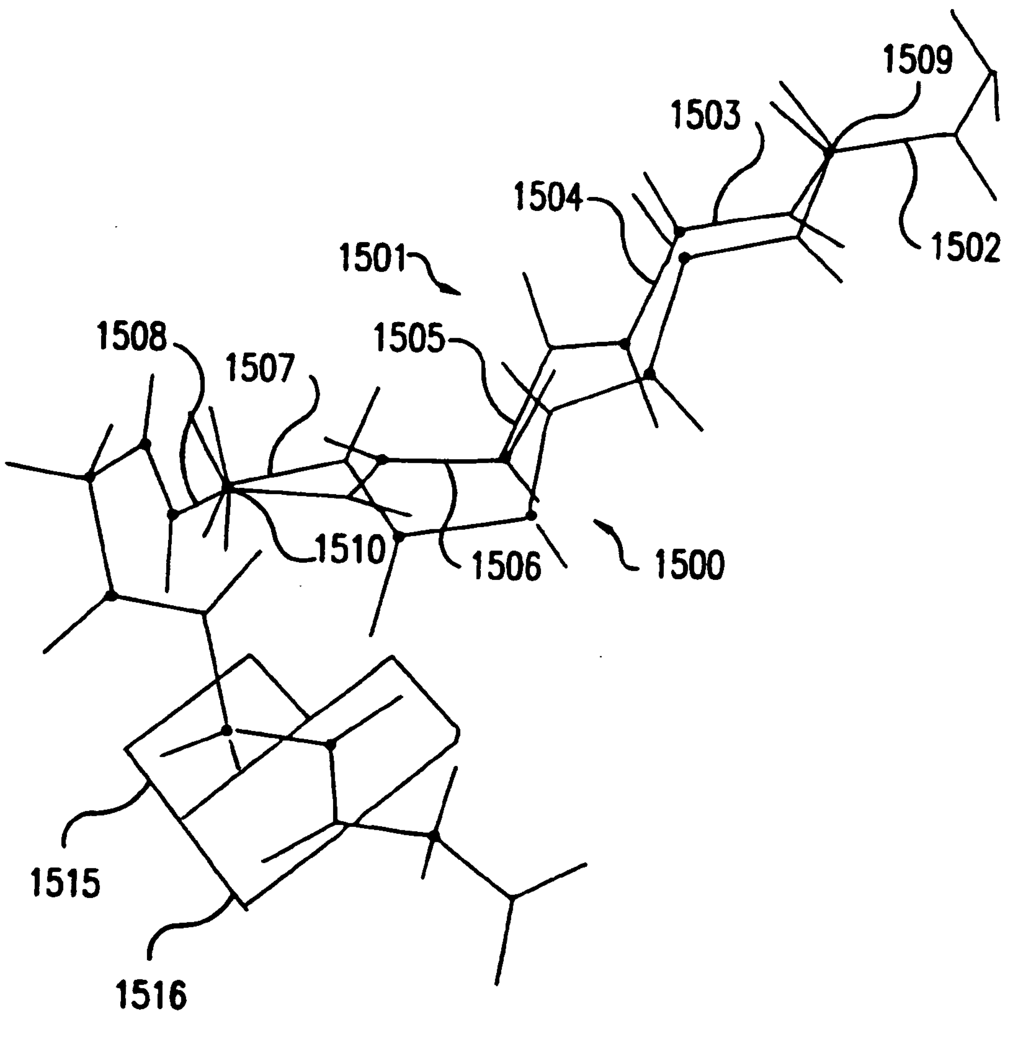

1509
1503
1504
1502
1501
1505
1508
1507
1510
1506
1500
1515
1516

FIG.13

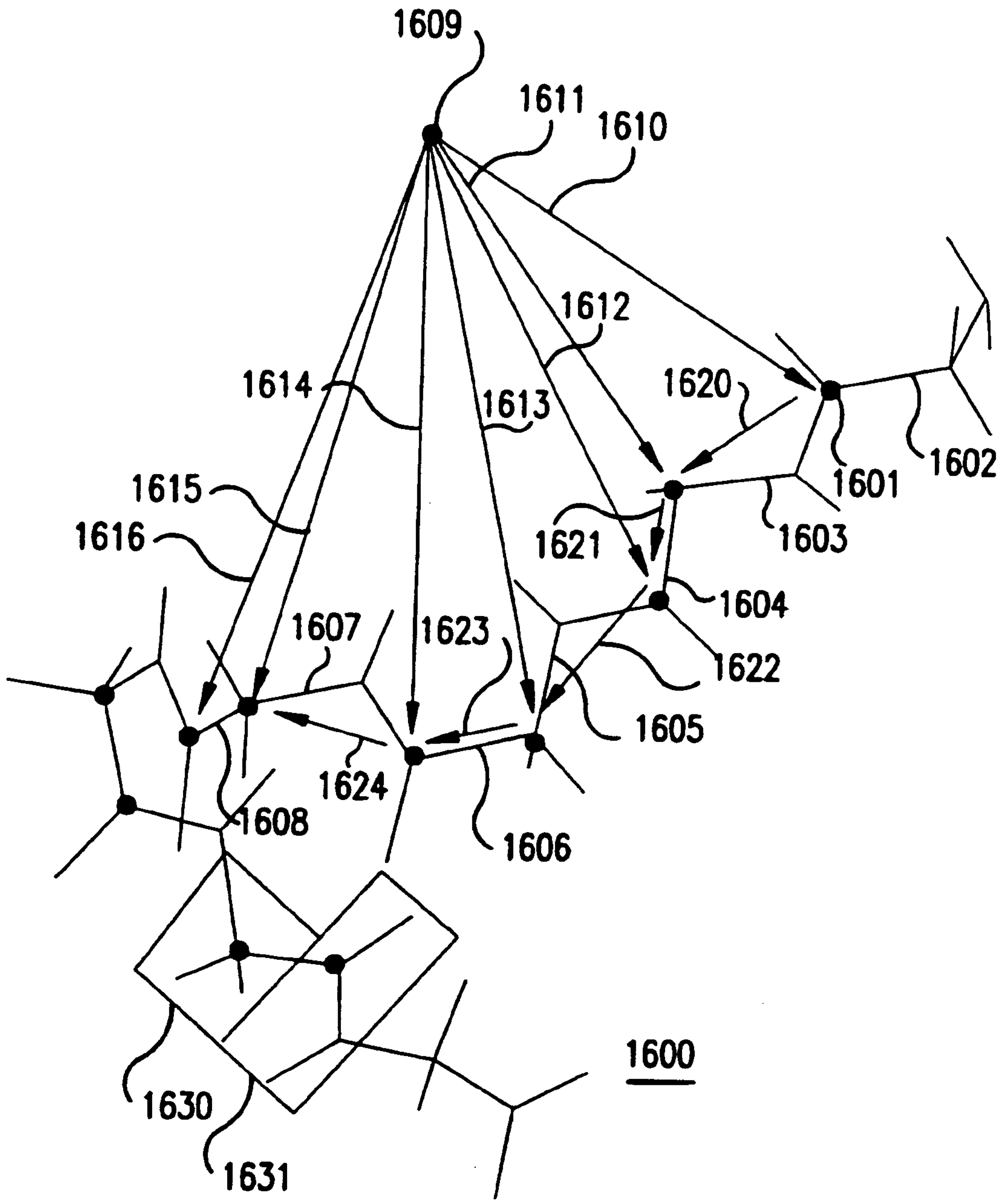

1609
1611
1610
1612
1620
1614
1613
1602
1601
1615
1603
1616
1621
1604
1607
1623
1622
1605
1624
1608
1606
1600
1630
1631

FIG.14

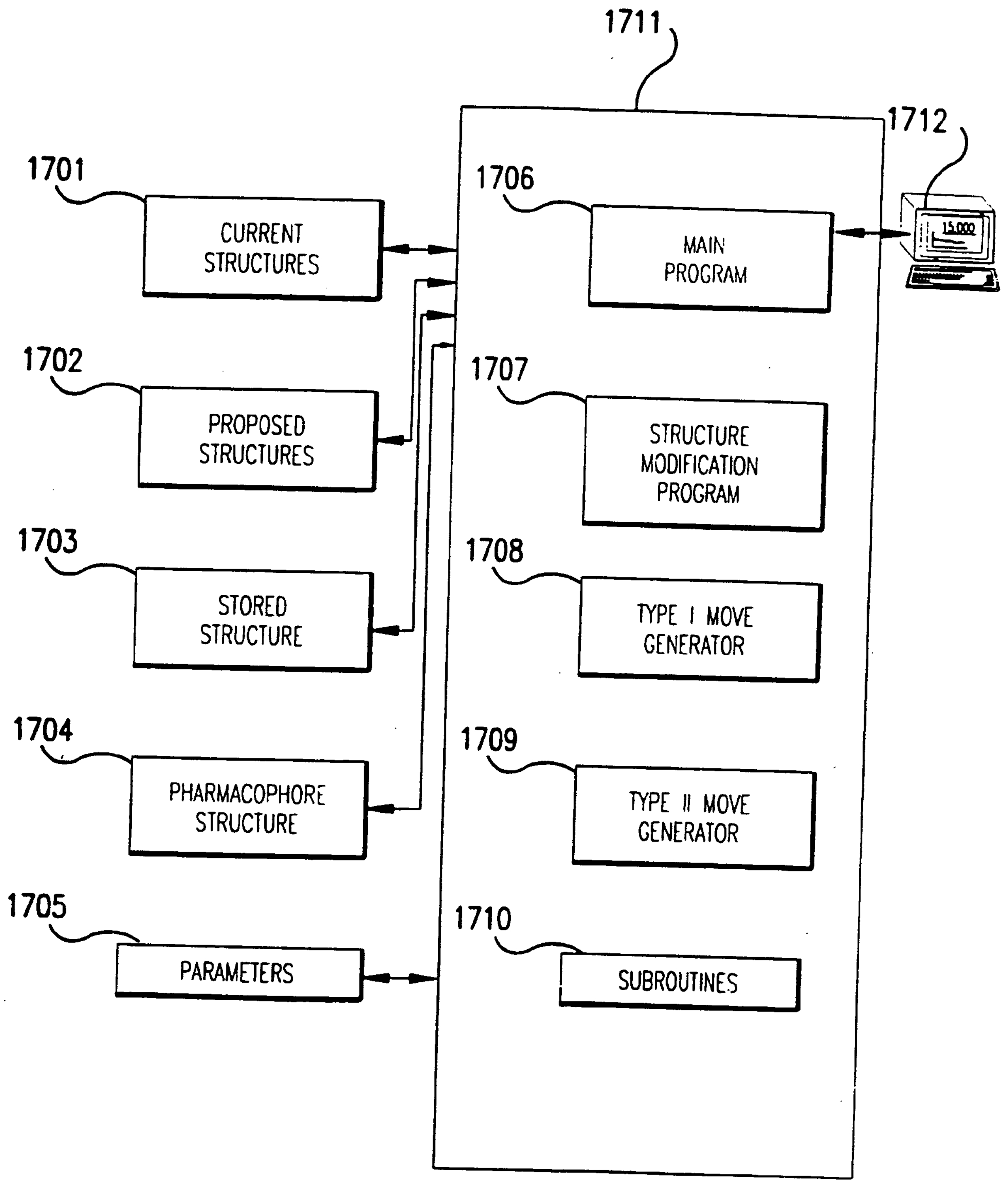

1711
1712
1701
CURRENT STRUCTURES
1702
PROPOSED STRUCTURES
1703
STORED STRUCTURE
1704
PHARMACOPHORE STRUCTURE
1705
PARAMETERS
1706
MAIN PROGRAM
1707
STRUCTURE MODIFICATION PROGRAM
1708
TYPE I MOVE GENERATOR
1709
TYPE II MOVE GENERATOR
1710
SUBROUTINES
15.000

FIG.15

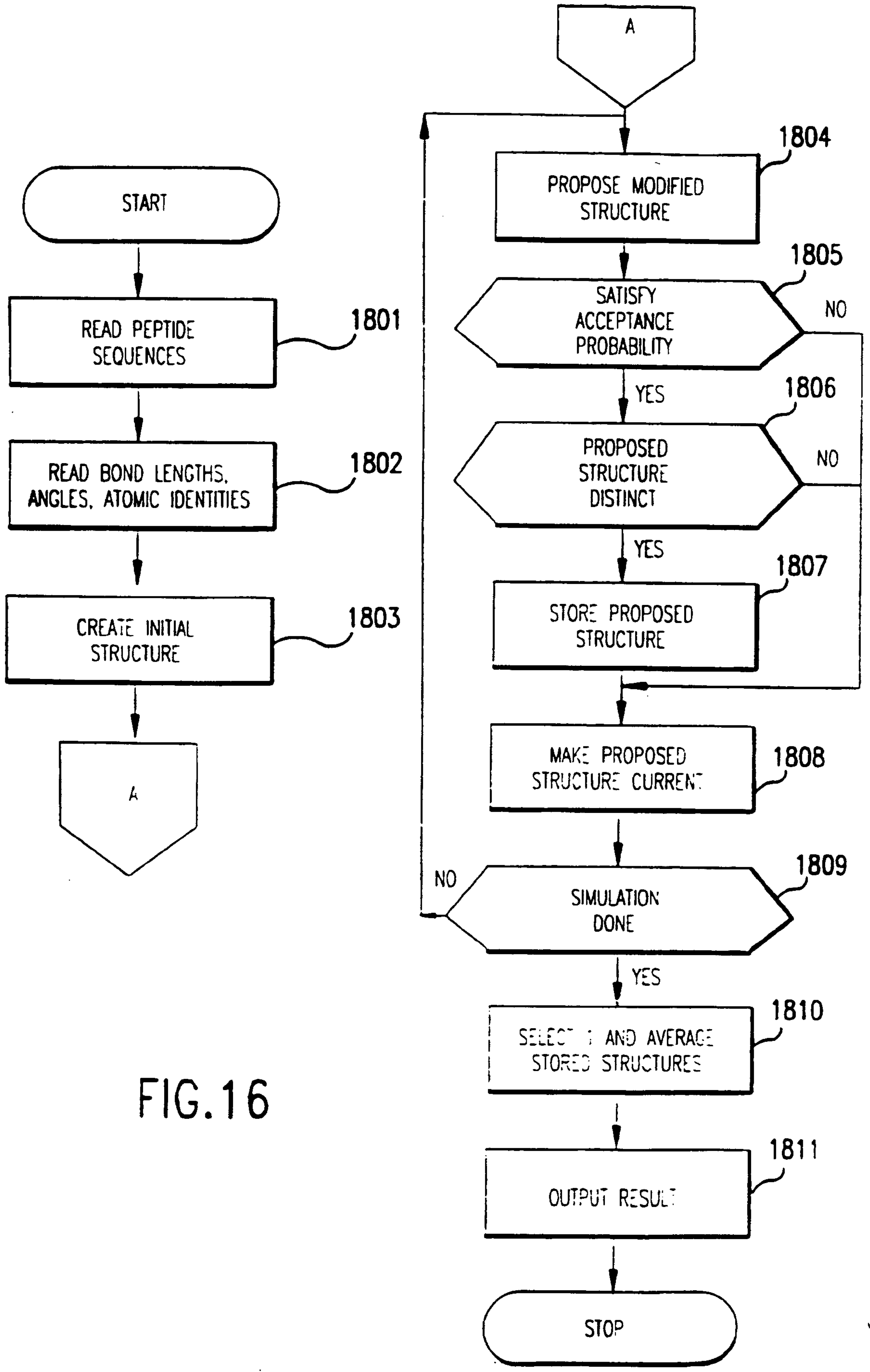

START
READ PEPTIDE SEQUENCES
1801
READ BOND LENGTHS, ANGLES, ATOMIC IDENTITIES
1802
CREATE INITIAL STRUCTURE
1803
A
A
PROPOSE MODIFIED STRUCTURE
1804
SATISFY ACCEPTANCE PROBABILITY
1805
NO
YES
PROPOSED STRUCTURE DISTINCT
1806
NO
YES
STORE PROPOSED STRUCTURE
1807
MAKE PROPOSED STRUCTURE CURRENT
1808
NO
SIMULATION DONE
1809
YES
SELECT AND AVERAGE STORED STRUCTURES
1810
OUTPUT RESULT
1811
STOP

FIG.16

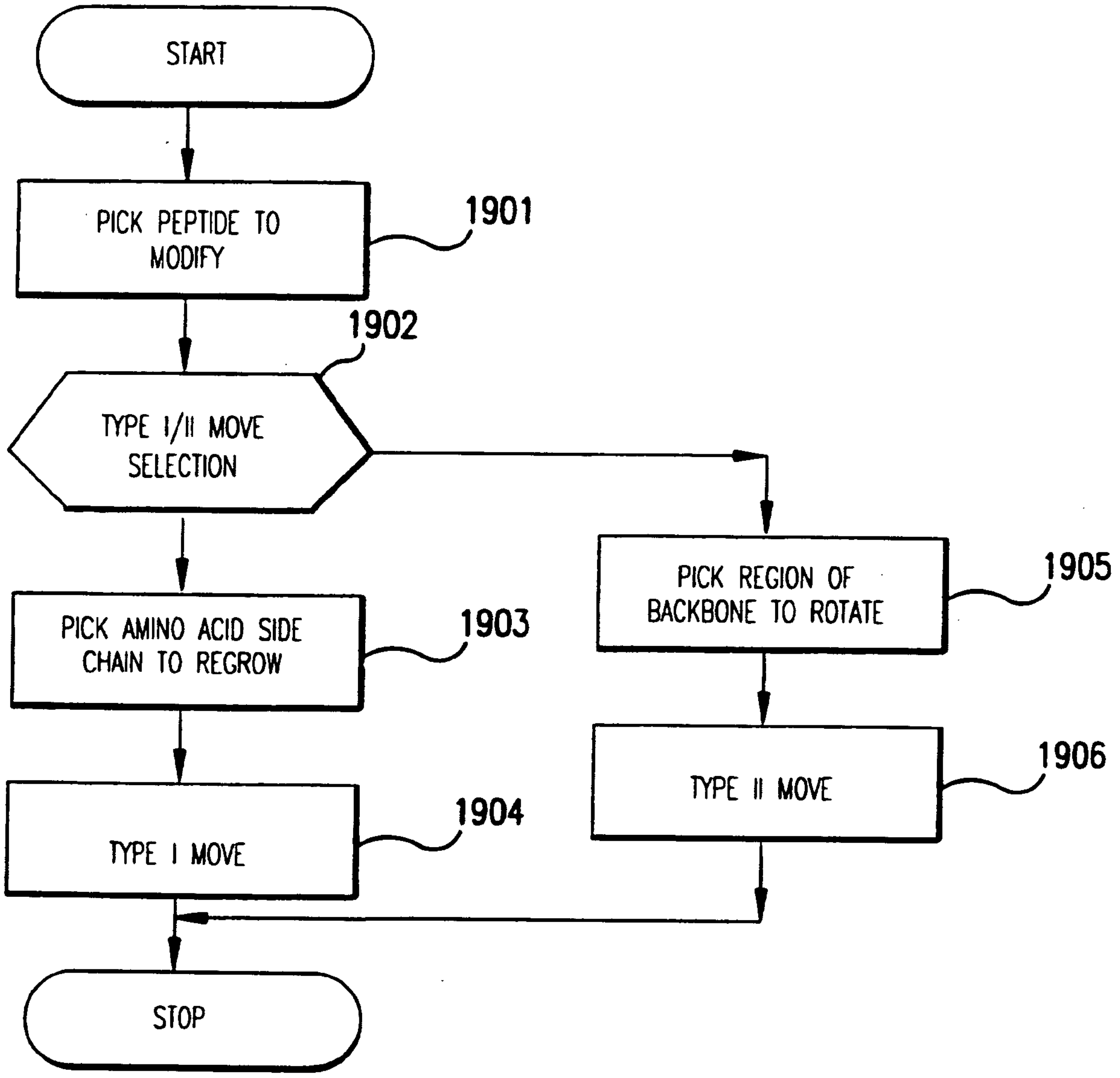

START
PICK PEPTIDE TO MODIFY
1901
1902
TYPE I/II MOVE SELECTION
PICK AMINO ACID SIDE CHAIN TO REGROW
1903
TYPE I MOVE
1904
PICK REGION OF BACKBONE TO ROTATE
1905
TYPE II MOVE
1906
STOP

FIG.17

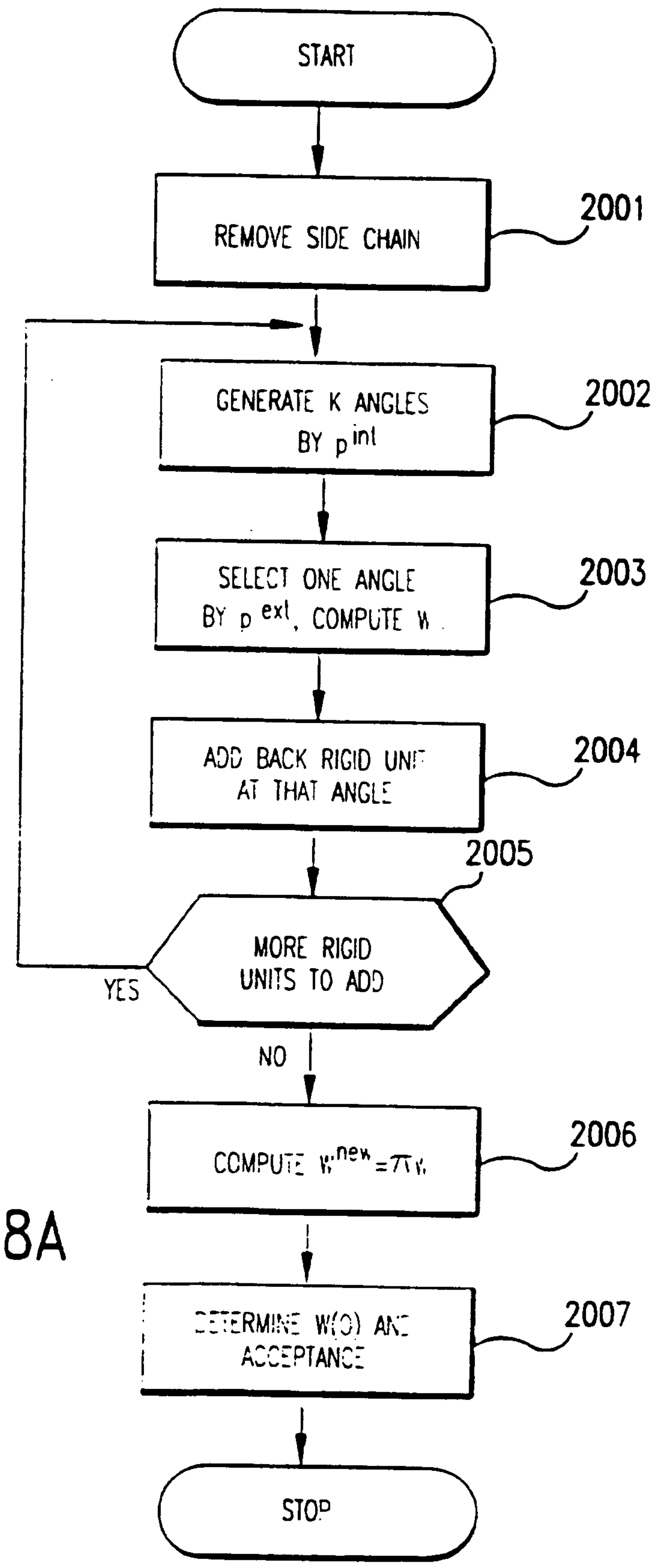

START
REMOVE SIDE CHAIN
2001
GENERATE K ANGLES BY p^{int}
2002
SELECT ONE ANGLE BY p^{ext}, COMPUTE W
2003
ADD BACK RIGID UNIT AT THAT ANGLE
2004
2005
MORE RIGID UNITS TO ADD
YES
NO
COMPUTE $W^{new} = \Pi W$
2006
DETERMINE W(O) AND ACCEPTANCE
2007
STOP

FIG.18A

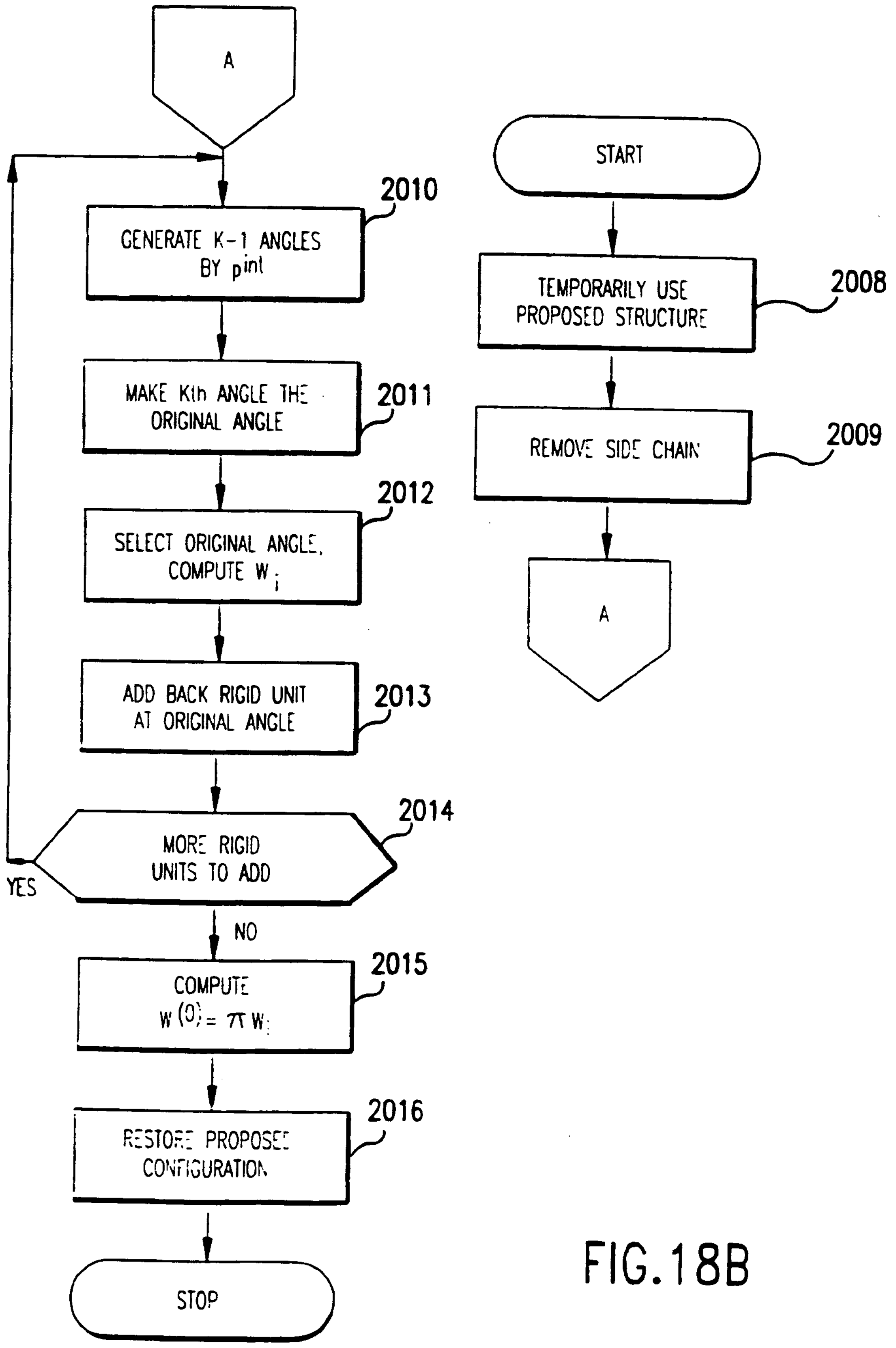

A
GENERATE K-1 ANGLES BY p^{int}
2010
MAKE Kth ANGLE THE ORIGINAL ANGLE
2011
SELECT ORIGINAL ANGLE, COMPUTE W_i
2012
ADD BACK RIGID UNIT AT ORIGINAL ANGLE
2013
MORE RIGID UNITS TO ADD
2014
YES
NO
COMPUTE $W^{(0)} = \pi W_i$
2015
RESTORE PROPOSED CONFIGURATION
2016
STOP
START
TEMPORARILY USE PROPOSED STRUCTURE
2008
REMOVE SIDE CHAIN
2009
A

FIG.18B

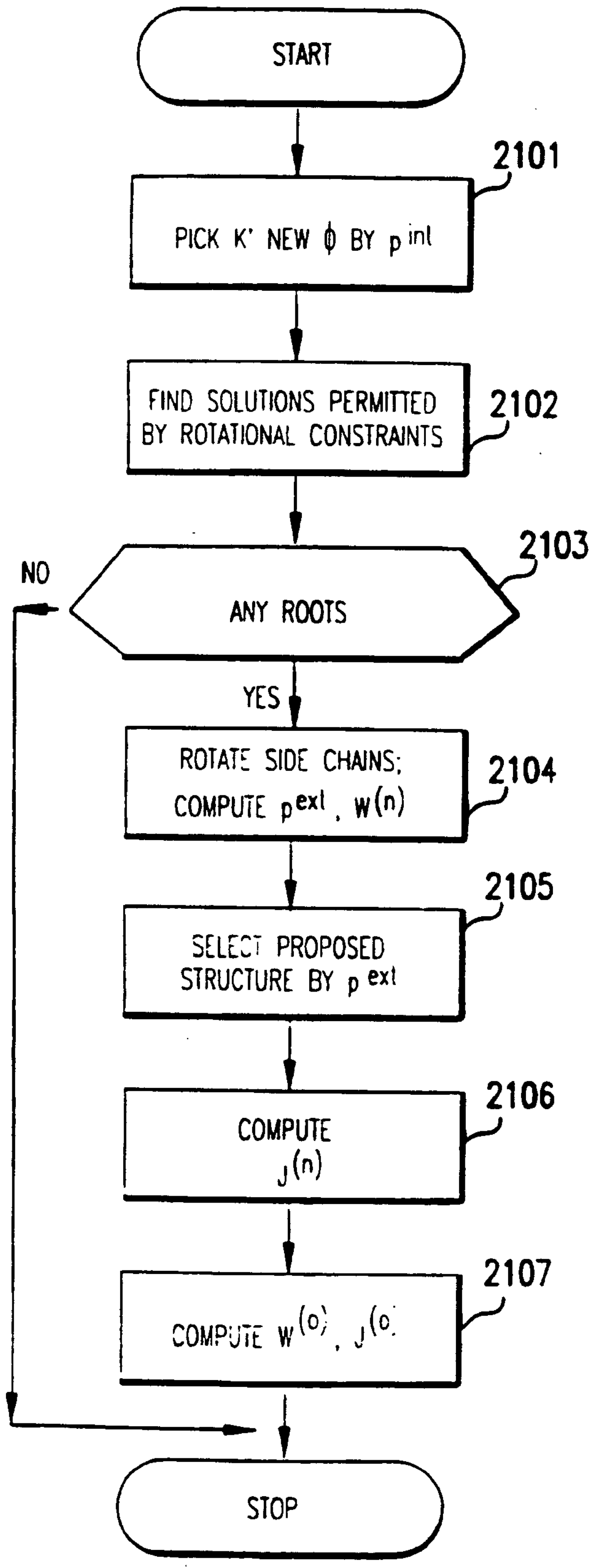

START
2101
PICK K' NEW ϕ BY p^{int}
FIND SOLUTIONS PERMITTED BY ROTATIONAL CONSTRAINTS
2102
ANY ROOTS
2103
NO
YES
ROTATE SIDE CHAINS; COMPUTE p^{ext}, $w^{(n)}$
2104
SELECT PROPOSED STRUCTURE BY p^{ext}
2105
COMPUTE $J^{(n)}$
2106
COMPUTE $w^{(o)}$, $J^{(o)}$
2107
STOP

FIG.19A

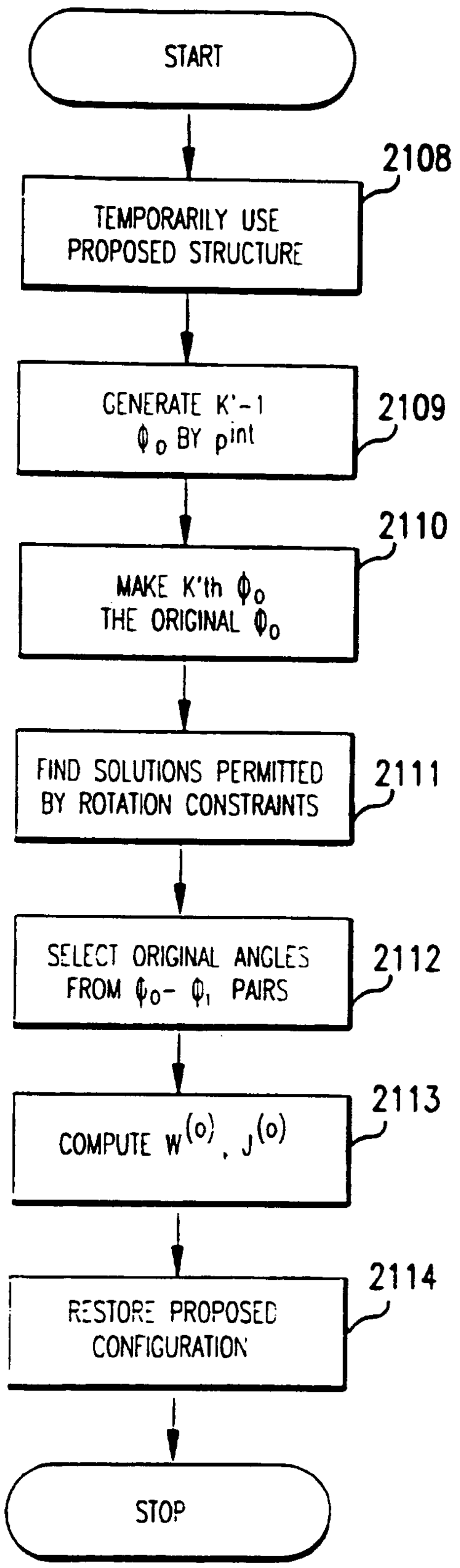

START
TEMPORARILY USE PROPOSED STRUCTURE
2108
GENERATE K'-1 ϕ_0 BY p^{int}
2109
MAKE K'th ϕ_0 THE ORIGINAL ϕ_0
2110
FIND SOLUTIONS PERMITTED BY ROTATION CONSTRAINTS
2111
SELECT ORIGINAL ANGLES FROM ϕ_0- ϕ_1 PAIRS
2112
COMPUTE $w^{(o)}$, $J^{(o)}$
2113
RESTORE PROPOSED CONFIGURATION
2114
STOP

FIG.19B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/50, 159:00; G06G 7/58; G01N 24/12, 33/53
US CL :364/413.01, 496, 578; 436/173, 501, 518; 435/7.1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/413.01, 496, 578; 436/173, 501, 518; 435/7.1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, STN
search terms: library, pharmacophore, REDOR NMR, monte carlo

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- A	HODGKIN et al. A Monte Carlo Pharmacophore Generation Procedure: Application to the Human PAF Receptor. Journal of Computer Aided Molecular Design. 1993, Vol. 7, pages 515-534, see pages 517-521.	22-24, 26, 27, 51-106 --- 1-21, 25, 28-41, 108
X --- A	WILSON et al. The Calculation and Synthesis of a Template Molecule. Tetrahedron. 1993, Vol. 49, No. 17, pages 3655-3663, see entire document.	51-106 --- 1-41, 108
P, A	SEPETOV et al. Library of libraries: Approach to synthetic combinatorial library design and screening of "pharmacophore" motifs. Proceedings of the National Academy of Sciences. June 1995, Vol. 92, pages 5426-5430, see abstract.	1-41, 108

[✓] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:
"A"	document defining the general state of the art which is not considered to be of particular relevance
"E"	earlier document published on or after the international filing date
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O"	document referring to an oral disclosure, use, exhibition or other means
"P"	document published prior to the international filing date but later than the priority date claimed
"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&"	document member of the same patent family

Date of the actual completion of the international search 03 JULY 1996	Date of mailing of the international search report 25 JUL 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer [illegible] LORA M. GREEN Telephone No. (703) 308-0196

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	BIANCHI et al. A Conformationally Homogenous Combinatorial Peptide Library. Journal of Molecular Biology. 1995, Vol. 247, pages 154-160, see abstract.	1-41, 108
X	US 5,265,030 (SKOLNICK ET AL.) 23 November 1993, see column 2, line 21-column 3, line 20.	68-102, 104-106
Y	GARBOW et al. Determination of the Molecular Confirmation of Melanostatin Using 13C,15N-REDOR NMR Spectroscopy. Journal of the American Chemical Society. 1993, Vol. 115, pages 238-244, see Experimental Section.	42-50, 107
Y	FERNANDEZ et al. Magnetic Resonance Studies of Polypeptides Adsorbed on Silica and Hydroxyapatite Surfaces. Journal of the American Chemical Society. 1992, Vol. 114, pages 9634-9642, see abstract.	42-50, 107

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Please See Extra Sheet.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest ☐ The additional search fees were accompanied by the applicant's protest.
☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/04229

BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING
This ISA found multiple inventions as follows:

This application contains the following inventions or groups of inventions which are not so linked as to form a single inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fees must be paid.

Group I, claim(s) 1-41 and 108, drawn to a method of determining a consensus pharmacophore structure.
Group II, claim(s) 42-50 and 107, drawn to a method of making solid state magnetic resonance methods.
Group III, claim(s) 51-67 and 103, drawn to a method of configurational Monte Carlo determination.
Group IV, claims 68-102 and 104-106, drawn to an apparatus for configurational bias Monte Carlo determination.

The inventions listed as Groups I-IV do not relate to a single inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons:

Groups I and Groups II and III do not share a special technical feature as each has different steps and different end results. The method of Group I requires screening of a diversity library and generation of a pharmacophore, neither of which is required by the methods of Groups II and III. The distances required for the pharmacophore generation of Group I could be obtained by methods other than through the use of solid-state NMR methods ofGroup II, such as by solution NMR or X-ray crystallography. In addition, the method of Group I as claimed does not require the method of Group III as claimed, as the dependant claim of Group I which specifies the Monte Carlo method (claim 24) requires generating a proposed structure based on data for diversity libraries, whereas the method of Group II does not require the use of diversity libraries. Thus, Groups II and III lack the special technical feature of Group I, i.e. using a diversity library to generate a consensus pharmacophore structure.

Groups II and III are related as separate methods, as Group II is drawn to a method of making NMR measurements, while the method of Group III is drawn to a method of configurational monte carlo analysis. Thus, Groups II and III do not share a technical feature.

Groups I and II are related to Group IV as separate methods and product, as the methods of Groups I and II as claimed do not require the apparatus of Group IV as claimed.

Groups III and IV are related as separate method and product. The method of Group III as claimed does not require the use of the apparatus of Group IV as claimed. In addition, the apparatus of Group IV could be used in methods other than the method of Group III such as use generating structures using NMR data obtained from a compound in solution phase.